NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MVC FOR CONTENT MANAGEMENT ON THE CLOUD

by

Crystal A. McGruder

September 2011

Thesis Advisor:	Doron Drusinsky
Co-Advisor:	Man-Tak Shing

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2011	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE MVC for Content Management on the Cloud	5. FUNDING NUMBERS
6. AUTHOR(S) Crystal A. McGruder	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Cloud computing portrays a new model for providing IT services over the Internet. In cloud computing, resources are accessed from the Internet through web-based tools. Although cloud computing offers reduced cost, increased storage, high automation, flexibility, mobility, and the ability of IT to shift focus, there are other concerns—such as the management, organization and structure of content on the cloud—that large organizations should consider before migrating to the cloud. This thesis presents an overview of Model View Controller (MVC) architectural pattern and describes how the pattern can be applied to the cloud for content management. The MVC architecture is proposed in this thesis because it divides the aspects of a document into three parts: a model, view and controller, thus allowing elasticity, portability, and interoperability for document objects. The thesis presents a case study to illustrate how MVC can be used to facilitate document collaboration and content management in the cloud, and examines existing document standards to assess their readiness in supporting the MVC document architecture.

14. SUBJECT TERMS Cloud Computing, Model View Controller, Open Office XML, OpenDocument Format, Rich Text Format	15. NUMBER OF PAGES 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

MVC FOR CONTENT MANAGEMENT ON THE CLOUD

Crystal A. McGruder
Civilian, United States Navy
B.S., University of Arkansas at Pine Bluff, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2011

Author: Crystal A. McGruder

Approved by: Doron Drusinsky
Thesis Advisor

Man-Tak Shing
Co-Advisor

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Cloud computing portrays a new model for providing IT services over the Internet. In cloud computing, resources are accessed from the Internet through web-based tools. Although cloud computing offers reduced cost, increased storage, high automation, flexibility, mobility, and the ability of IT to shift focus, there are other concerns—such as the management, organization and structure of content on the cloud—that large organizations should consider before migrating to the cloud.

This thesis presents an overview of Model View Controller (MVC) architectural pattern and describes how the pattern can be applied to the cloud for content management. The MVC architecture is proposed in this thesis because it divides the aspects of a document into three parts: a model, view, and controller, thus allowing elasticity, portability, and interoperability for document objects. The thesis presents a case study to illustrate how MVC can be used to facilitate document collaboration and content management in the cloud, and examines existing document standards to assess their readiness in supporting the MVC document architecture.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I would like to thank God for bringing me through this interesting experience. He answered all my prayers and encouraged me when I felt like I wanted to give up. God has been so amazing to me and I want to dedicate not only my life but this thesis to HIM. I would be nothing without HIM and HIS son Jesus. Thank YOU, Lord, for being so faithful to me!

Next, I would like to thank my loving husband, Robert. He is truly the best! I am so grateful to have him in my life. He has been so patient with me and I love him for that. Robert spent many nights listening to me read and type on my laptop. There were days when he would cook dinner and clean the house for me so that I could focus on my research. I feel like he wrote this thesis with me. He has supported me not only with this thesis but throughout this whole program. I am the luckiest woman in the world because I have him.

Next, I also would like to thank management for believing in me, supporting me and giving me the opportunity to accomplish my goals. And I want to thank two of my "special co-workers" for all of their encouragement. Their love and inspirations inspired me to try harder. They listened to ALL of my complaints and gave me wonderful advice. I appreciate them very much.

And finally, I MUST thank two of the BEST professors at NPS, Professor Drusinsky and Professor Shing. I am so grateful for ALL their help! They both gave me the direction and guidance that I need to reach this difficult goal. I am truly blessed to have them as my thesis and co-thesis advisors. Their incredible knowledge assisted me throughout this whole process. There was a time that I felt frustrated about my slow progression in my thesis research and Professor Drusinsky sent me a very encouraging e-mail; whenever I felt like I giving up, I would read his e-mail and it gave me that extra push. I also remember when I felt like I did not have anything else to contribute to this thesis and Professor Shing gave me a list of other helpful topics and questions to research. Thank you BOTH for everything!

THIS PAGE INTENTIONALLY LEFT BLANK

I. CLOUD COMPUTING

A. BACKGROUND

In the past, a drawing of network was illustrated as a cloud. The cloud was used as a metaphor for the Internet; when combining *cloud* with *computing*, the definition expands. The term *cloud computing* was first used in a lecture in 1997 by Ramnath Chellappa, an Associate Professor in the Information Systems & Operations Management. Chellappa defined cloud computing as the "new computing paradigm where the boundaries of computing will be determined by economic rationale instead of technical limits alone." In 1999, Salesforce.com was the first to move into cloud computing, next was Amazon through Amazon Web Service in 2002, then Google Docs in 2006.

Cloud computing is the fifth generation of computing that provides five essential characteristics (*on-demand self-services, resource pooling, measured services, board network, and rapid elasticity*), three service models (*Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS)*), and four deployment models (*private cloud, community cloud, public cloud, and hybrid cloud*). It is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell & Tim, 2009). Figure 1 gives examples of the three deployment models.

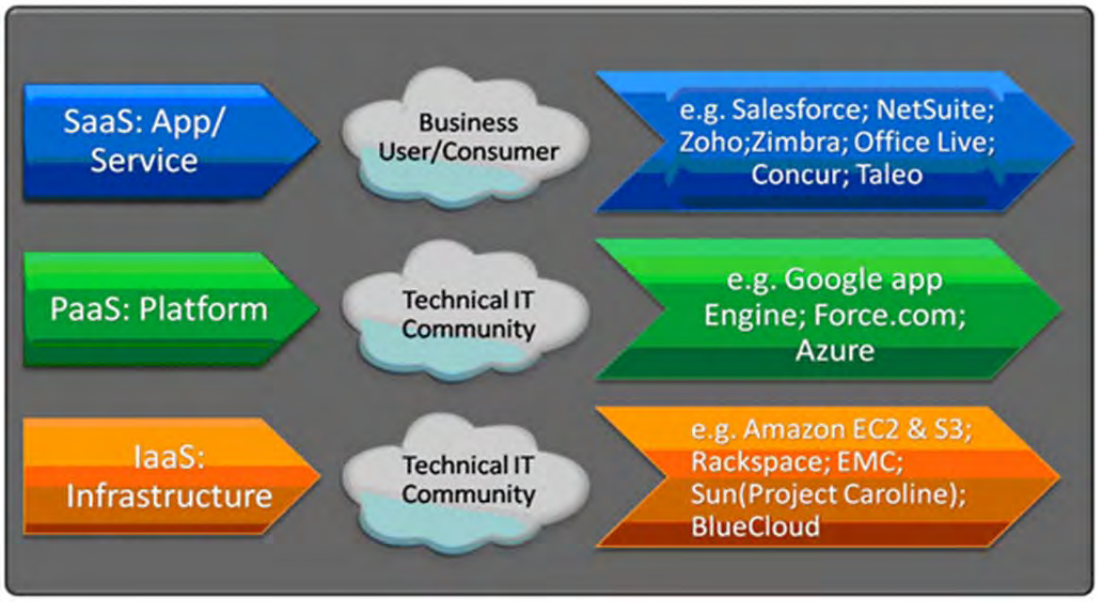


Figure 1. Cloud Computing Deployment Models (From Shum, 2009)

Cloud computing offers many benefits to organizations such as Department of Defense. When the Department of Defense prepares to migrate to the cloud, it should consider how the content in its documents would be managed over time. Managing content in documents that requires logic can make documents difficult to manage and keep organized. For an example, a company has tax report documents that include state taxes and county fees; when these tax rates are deeply embedded in each document, it makes the content of these documents very difficult to manage because users are required to update these documents whenever the state or county changes its rates. Managing content in these documents can become cumbersome.

The difficulty of the managing content in documents is decreased when the Model View Controller (MVC) approach is applied to separate the model, view and controller aspects of a document. For instance, applying the MVC approach to the tax report documents will separate the controller (tax rates calculation) from the view (display) of the document; now the user only need to update the rates via the controller whenever they are changed, and the document will automatically update its display to show the correct state taxes and county fees throughout the document.

B. THESIS OBJECTIVES

The objective of this thesis is to investigate how the Model View Controller (MVC) architecture can be used to address the content management risk in cloud computing. In this thesis, we will introduce the MVC pattern and illustrate how it pertains to the content management, give an overview of the MVC architecture, discuss the distinct elements of MVC, provide a class diagram for the MVC pattern, and discuss the benefits of applying MVC to content management in the cloud. There are four examples of MVC applications presented in this thesis to illustrate how MVC has already been applied to other applications. We will also present a case study to illustrate how MVC can be used to facilitate document collaboration and content management in the cloud. To give a detailed understanding of the cloud documents, a use case analysis, requirements and class diagram are presented. There exist some document standards that can potentially be used to apply MVC to a document; we will examine three different document standards and evaluate their readiness to support MVC.

C. ORGANIZATION OF THESIS

The remainder of this thesis is organized as follows:

Chapter II provides an overview the MVC architectural pattern. It presents the three elements of the MVC pattern and explains how the three elements work together, and provides examples to show how MVC was used in existing applications in the cloud.

Chapter III discusses the benefits of applying MVC to content management in the cloud, describes a use case to illustrate how MVC can be used to manage cloud documents contents, and presents the requirements for a MVC document architecture.

Chapter IV reviews existing document standards and provides an assessment of their readiness in supporting the MVC document architecture.

Chapter V summarizes the thesis and discusses future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MODEL VIEW CONTROLLER

A. BACKGROUND

The Model View Controller (MVC) is a widely supported design pattern invented in 1979 at the XEROX Palo Alto Research Center by Trgve Reenskaug. Reenskaug summarizes his inspiration:

> One of the great inventions of the Smalltalk group at Xerox Palo Alto Research Center (PARC) in the seventies was the idea that objects can be made visible on the computer screen so that the user can see and manipulate them directly. This makes the abstract computer data appear concrete and the underlying object model visible. The user can easily adjust his mental model to this computer model and operate on it with confidence. The well-designed direct manipulation object interface is intuitively obvious and therefore easy to learn for the uninitiated. This strength of the direct manipulation object model is also its main weakness. Each object can only appear once on the screen and must always be presented in the same way to preserve the illusion of concreteness. This is insufficient for large and complex models where we need to view objects in different ways. (Pawson, 2004)

The original name was the "Thing Model View Editor pattern," then Reenskaug changed the name to "Model View Controller Pattern" (Walther, 2008). He sought after solving the problem of representing complex real-world systems such as "the design and construction of a major bridge, a power station or an off-shore oil production platform" (Walther, 2008).

In 1980s, MVC made its first appearance to the public in Smalltalk-80. Jim Althoff first implemented MVC for Smalltalk–80. Smalltalk is an object-oriented programming language and MVC was the major feature. MVC was developed in conjunction with Smalltalk. Originally, the MVC and Smalltalk-80 pattern was developed for the Dynabook Project (a portable personal information management tool) by Alan Kay (Sasine, 1995).

The objective of MVC is to develop applications in a modular way, support in the development of graphical user interface (GUI), and use object sharing to endorse

software reusability. Initially, MVC was developed to map the traditional input, processing, output roles into the GUI realm (Joomla, 2010).

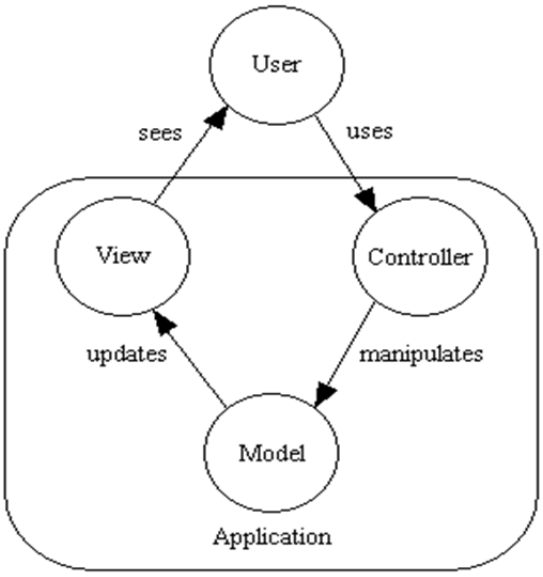


Figure 2. Traditional MVC Mapping. (From Joomla, 2010)

B. MODEL VIEW CONTROLLER ARCHITECTURE OVERVIEW

In the "Original MVC Pattern" (Figure 3), the view is not directly updated from the model. If the model is altered, an event raises and then the view is changed in response of the event. The controller modifies the model, and since the view is observing the model, the view gets updated (Walther, 2008).

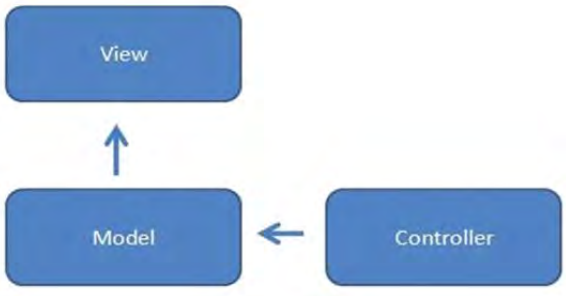


Figure 3. Original MVC Pattern. (From Walther, 2008)

MVC is normally associated with object-oriented frameworks; however, MVC is an architecture that can be implemented even without an object-oriented language or a specific class hierarchy (Wikipedia MVC: The Free Encyclopedia). The goal of the MVC architecture is to split the business logic and application data from the presentation data thus decreasing the complicated architectural design and increasing flexibility and maintainability of code. Matthew Cochran developed the following three diagrams to give an understanding of the MVC architecture. He demonstrates how the three elements work together and also how the end user interacts with a MVC based application.

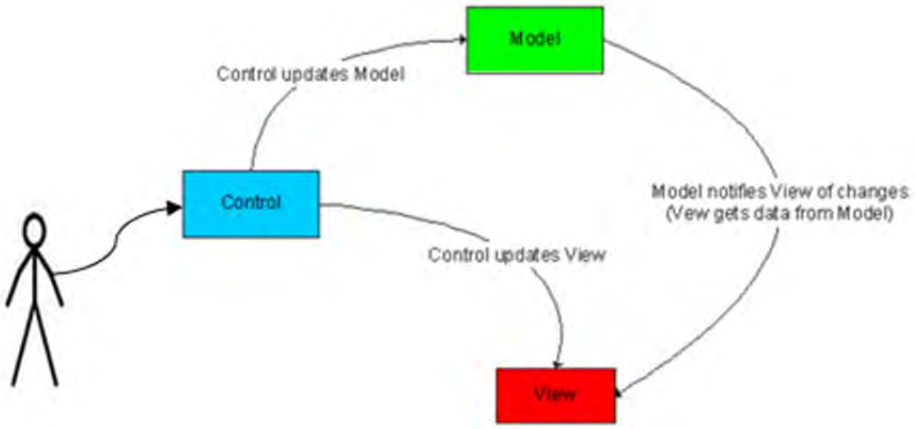


Figure 4. MVC Architecture Overview "A." (After Cochran, 2005)

Notice, in Figure 4, the controller makes a request to the model and updates the view. What happens if the end user wants to update the model using the view? See the next diagram.

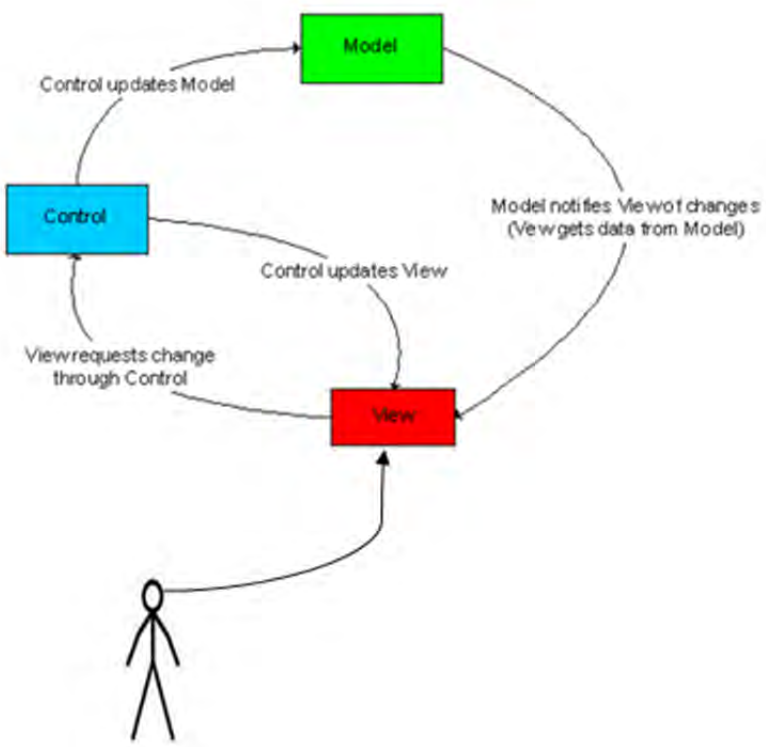


Figure 5. MVC Architecture Overview "B." (After Cochran, 2005)

Notice, in Figure 5, the end user makes a request using the control, through the view to update the model. Now, what happens if the view does not have the essential data to display the present state of the model?

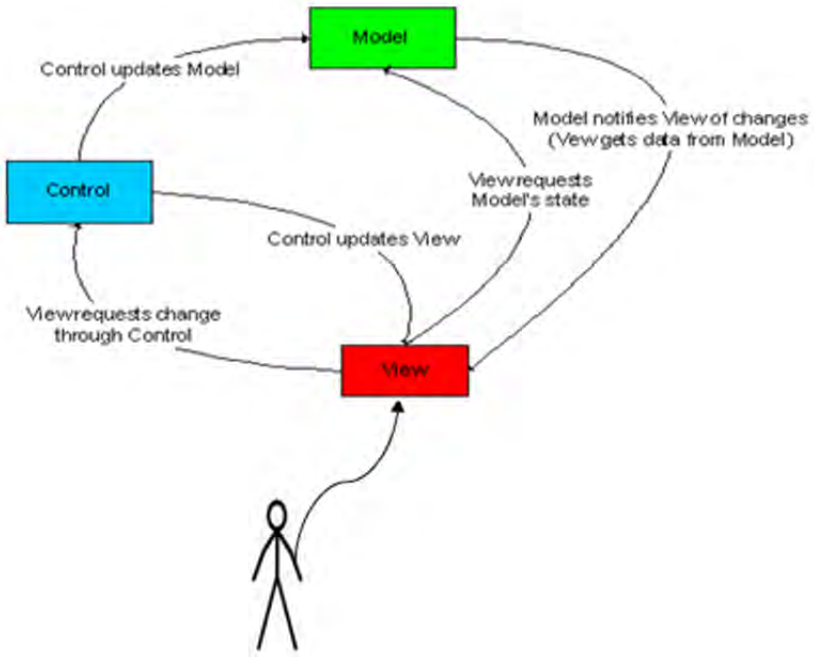


Figure 6. MVC Architecture Overview "C." (After Cochran, 2005)

Notice, in Figure 6, the view is enabled to get the state of the model in order to know what is needed to be displayed from the model. The end user uses the view to interact with the model. Typically, a request is started from the view then handled by the control; the control will then ask the model to change and make any necessary changes to the view (Cochran, 2005).

1. Model View Controller Distinct Elements

The Model View Controller architecture consists of three distinct elements: model, view, and controller. These distinct elements are separated to support maintaining and reusing code. Separation of these elements is used in applications that need to maintain multiple views of the same data. Figure 7 is a diagram that illustrates the relationships of the distinct elements.

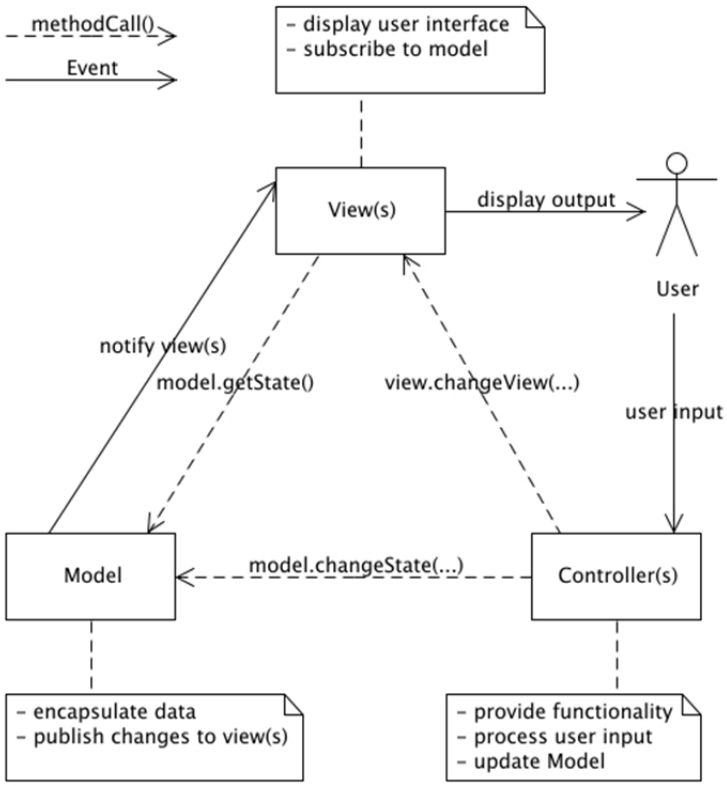


Figure 7. MVC Architecture of Component Relationship (From Gérardin, 2009)

a. View Element

Views control the display of data by displaying all or a fraction of the data and specifying how this data should be presented. Basically, it presents the data to the user. The view is an illustration of its model and data is received from the model by asking questions. The responsibilities of the view are to manage a section of the display and to keep the display consistent with the state of the model.

There are two types of views: Template View and Transform View. The template view is used to separate the HTML from the code, which uses a template to implement the view of the MVC. This template is typically a HTML document with embedded markers which are changed, manipulated, or evaluated by means of a template engine API to produce an output document (Battlez, 2007). The Transform View gets the model data one element at a time and transforms it into an end-user representation like HTML. The major difference between the two views is that a Template View is

organized around the output while a Transform view is organized around separate transforms for each kind of input element (Fowler, 2003).

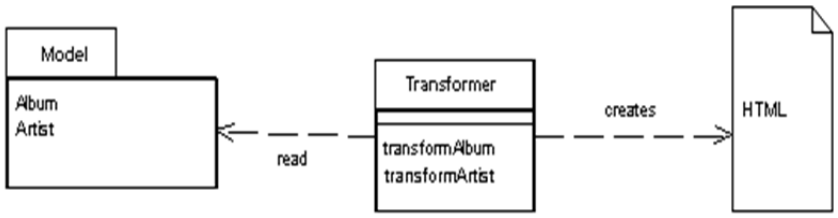


Figure 8. Transform View (From Fowler 2003)

b. Controller Element

The events affected by the model or view are handled by the controller. The controllers operates by accepting input from the user, then, based on the input, instructs the model and viewport to perform actions. The user and system is linked together through the controller. Basically, the controller handles user's input such as keystrokes, mouse clicks, and mouse movements as events. Sometimes the controller can be seen as the mediator between the view and the model; however, this is NOT the case because it does not sit between the model and view.

c. Model Element

The model represents the core functionality by managing the behavior and data of the application, providing requests with information about current state, and replying to instructions to alter the state. The model can be a single object or multiple objects. It is the data and business logic utilized in operating the data in an application in which making it more than just a database. For instance, not only does the model contain data but it also encapsulates methods to accessing and manipulating the data (data access layer).

The model element is also known as the domain logic. It can be built without any awareness of the views and controllers. This will allow the model to be independent of the view and the controller. However, it supplies services and data to the other elements of the MVC. The model is the core element of MVC.

There are two types of models: *passive model* and *active model*. In the passive model, a controller manipulates the model entirely. The web MVC commonly uses passive model. HTTP protocol is an example of passive model. In the passive model, first the controller modifies the model then informs the view that the model has changed and should be refreshed (MSDN).

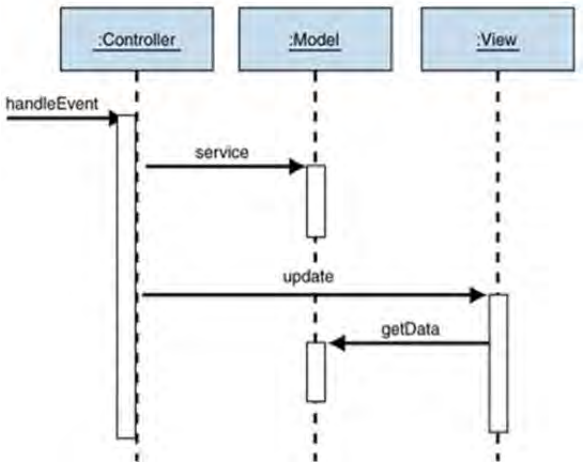


Figure 9. Behavior of the Passive Model (From MSDN)

The active model is the complete opposite of the passive model. In the active model, the model alters state without the controller's participation. This usually happens when other sources are changing the data and the changes must be reflected in the views (MSDN).

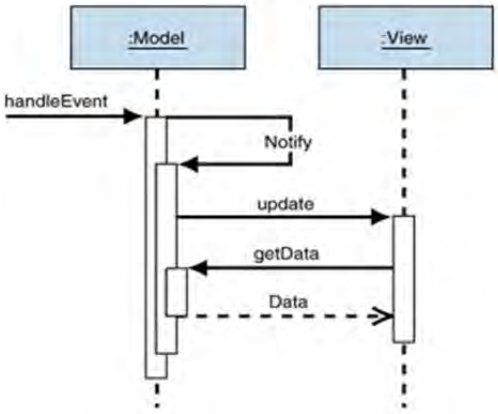


Figure 10. Behavior of the Active Model(From MSDN)

A passive model can be turn into an active model by using an adapter to add change propagation or other active model features to passive model objects (Battlez, 2007).

2. Class Diagram for the MVC Pattern

Usually, the model passes the data to controller then the controller passes that data to views. After receiving the data from the controller, the view will create a specific view to the end user. The controller helps the view to be apart from the model (Webworld). Below is a class diagram of a typical MVC application.

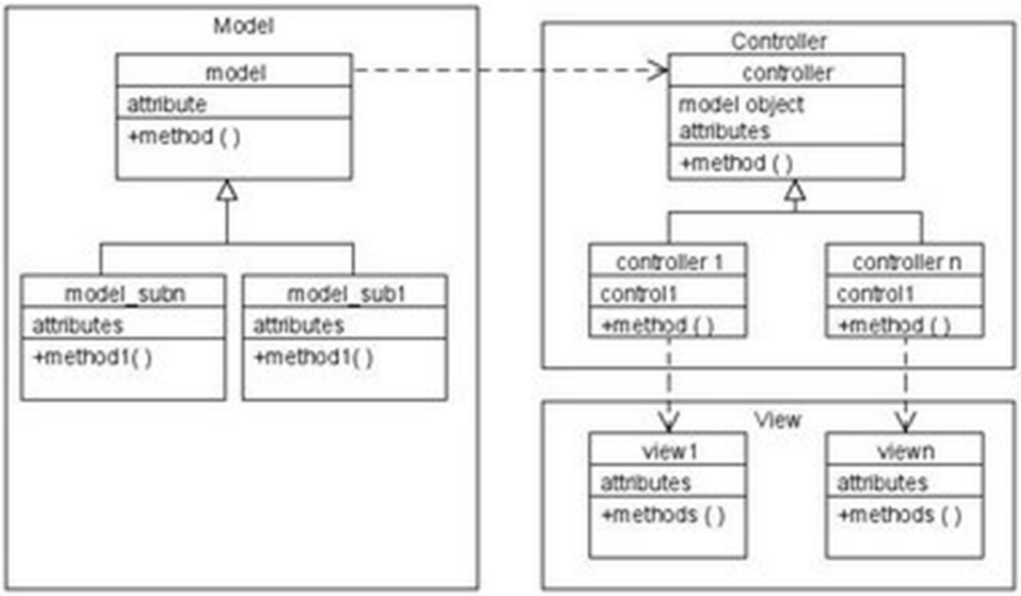


Figure 11. MVC Class Diagram (From Webworld, n.d.)

C. EXAMPLES OF MVC APPLICATIONS

1. Synchronized Disaster Response Version 1.0

In Shawn Kelly's thesis, he discusses how MVC is used in the Google Apps Framework. An application called Synchronized Disaster Response version 1.0 (SDR-1) is proposed in the thesis report. SDR-1 is a system designed to faster real-time information sharing openly and transparently across all levels of the public and private sectors, resulting in efficient use of resources during national disasters (Kelly & Mazyck,

2010). The application is divided into three distinct elements model, view, and controller. In the SDR-1 architecture, each distinct element is deployed to different physical servers in the cloud computing environment.

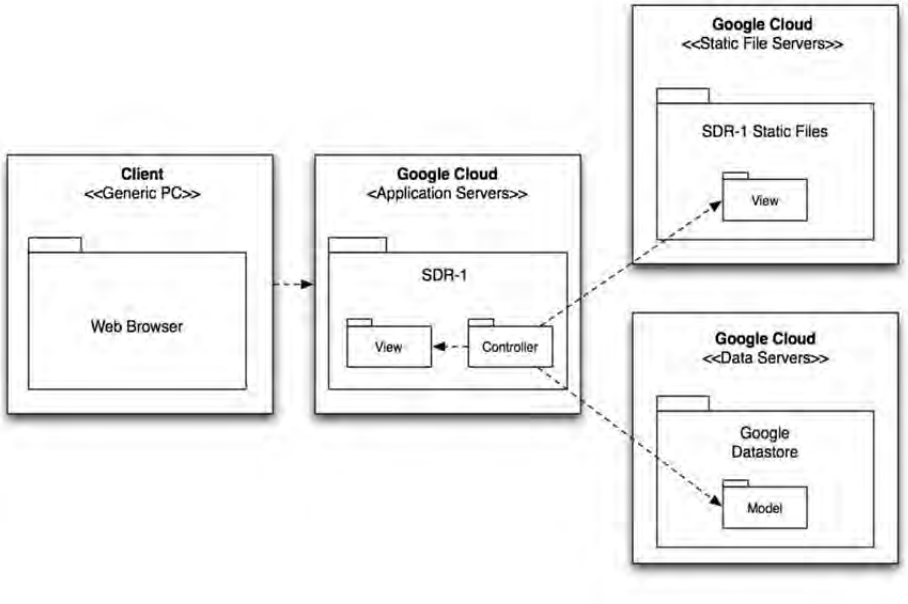


Figure 12. SDR Architecture (From Kelly & Mazyck, 2010)

The model consists of a set of persistent data in a datastore, which is a complex distributed data storage based on Google Bigtable. The Google Bigtable is a distributed, persistent, multi-dimensional sorted map that allows you to store strings (or un-interpreted array of bytes) indexed by row key, column key and timestamp. Seven classes are proposed in the SDR-1 application:

a. User Class – users across the Google ecosystem
b. Profile Class – additional user information
c. Organization Class – public & private organizations
d. Location Class – geographic position and its associated resources
e. Resource Class – information describing resources in abstract terms
f. Disaster Class – geographical area designated by FEMA/USSNORTHCOM after a disaster
g. Search Class – set of search criteria

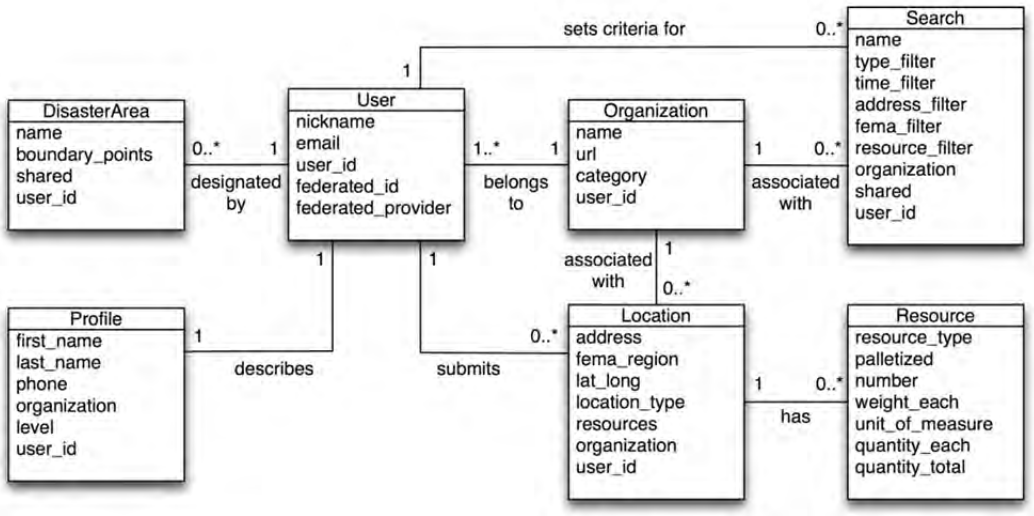


Figure 13. Data Model (From Kelly & Mazyck, 2010

The view is a set of HTML templates, CSS, and JavaScript used to define the look of the SDR-1 application. A template engine is also used in this application to "*merge data retrieved from the model with appropriate HTML templates to generate a view that can be returned to the user.*" The view's structure is dependent upon HTML templates. The templates describe content shown to the user via hypertext markup language and enables division of content into a hierarchy of extendable and reusable templates.

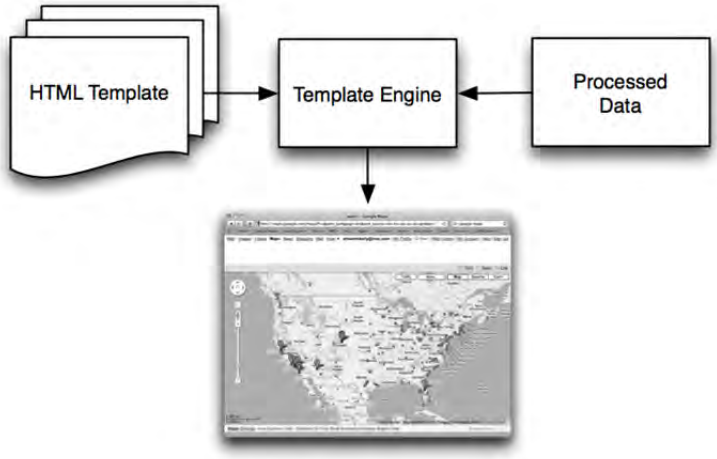


Figure 14. Template Engine (From Kelly & Mazyck, 2010)

The controller for the SDR-1 application has all of the business logic. The controller consists of configuration file and request handlers (Figure 15).

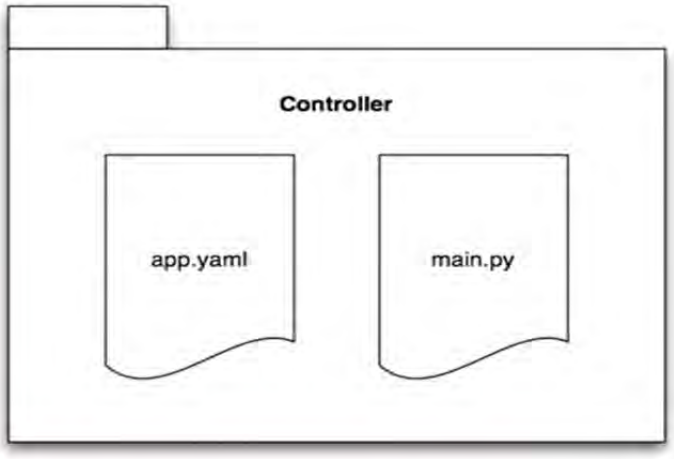


Figure 15. Controller (From Kelly & Mazyck, 2010)

The configuration file, app.yaml, "*provides a route to the appropriate request handler based on URL paths included in a request.*" The incoming HTTP requests for different URLs in application are dealt with by the request handlers in main.py. SDR-1 application contains Main Handler that extends a request handler, as shown in Figure 16.

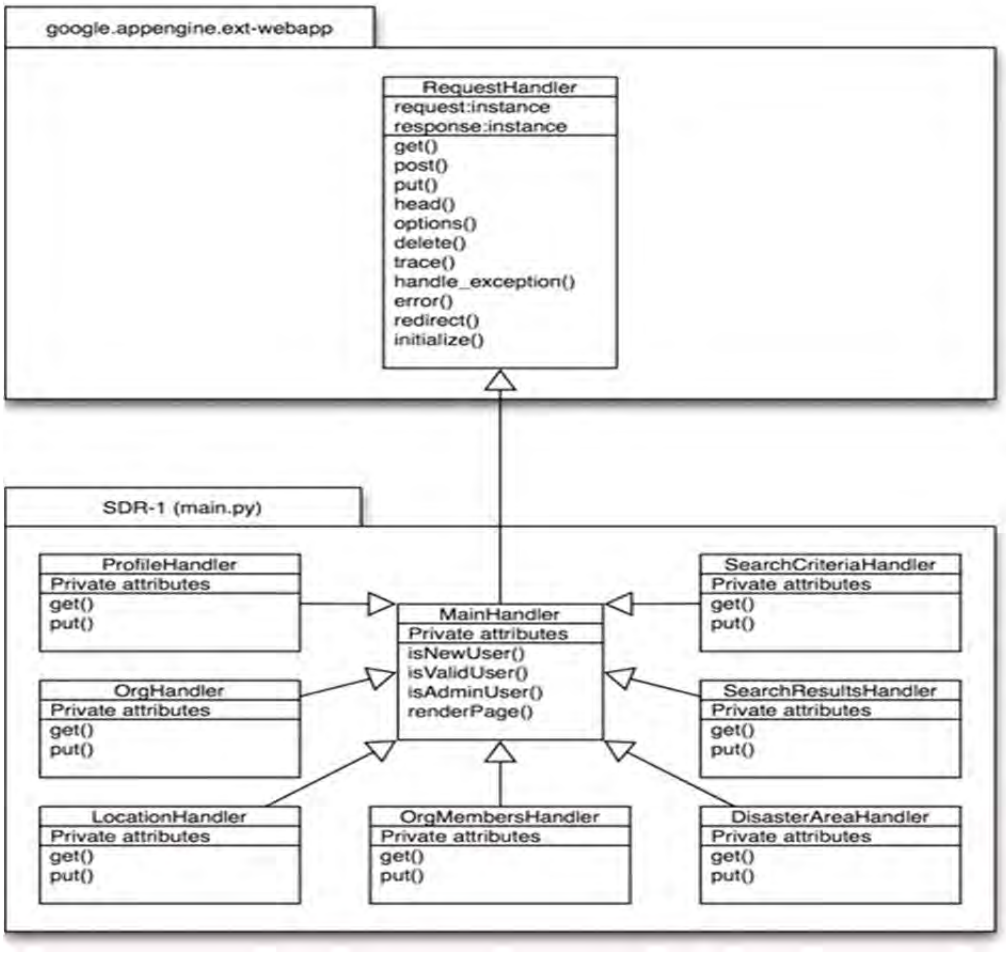


Figure 16. Event Handlers (From Kelly & Mazyck, 2010)

2. Information Service System of Cooperative Marketing of Tobacco Industry

Wang Bin, Bao Liwei, and Ye Yong did a case study on a system based on the SaaS model. This system is an information service system for cooperative marketing of tobacco industry. The system is divided into modules:

a. Statistical statement
b. Comprehensive inquiry
c. Product demonstrations
d. Market analysis
e. Dynamic monitoring

In this information service system for cooperative marketing of tobacco industry, the user can access his or her data storage organizations and give system access to other end-users. The user's configuration carries out the SaaS applications. Some users require more comprehensive information, such as the users from State Tobacco Monopoly Administration, which requires industry-wide marketing information for cigarette manufacturers and commercial enterprise to carry out the macro decision-making (Bin, Liwei, & Yong, 2009). Users of various cigarette manufacturers may need to view their products in the country's situation of production, sale and inventory. Notice, in Figure 17, the information service system for cooperative marketing of tobacco industry based on SaaS is in the maturity Ⅳ SaaS (Bin, Liwei, & Yong, 2009).

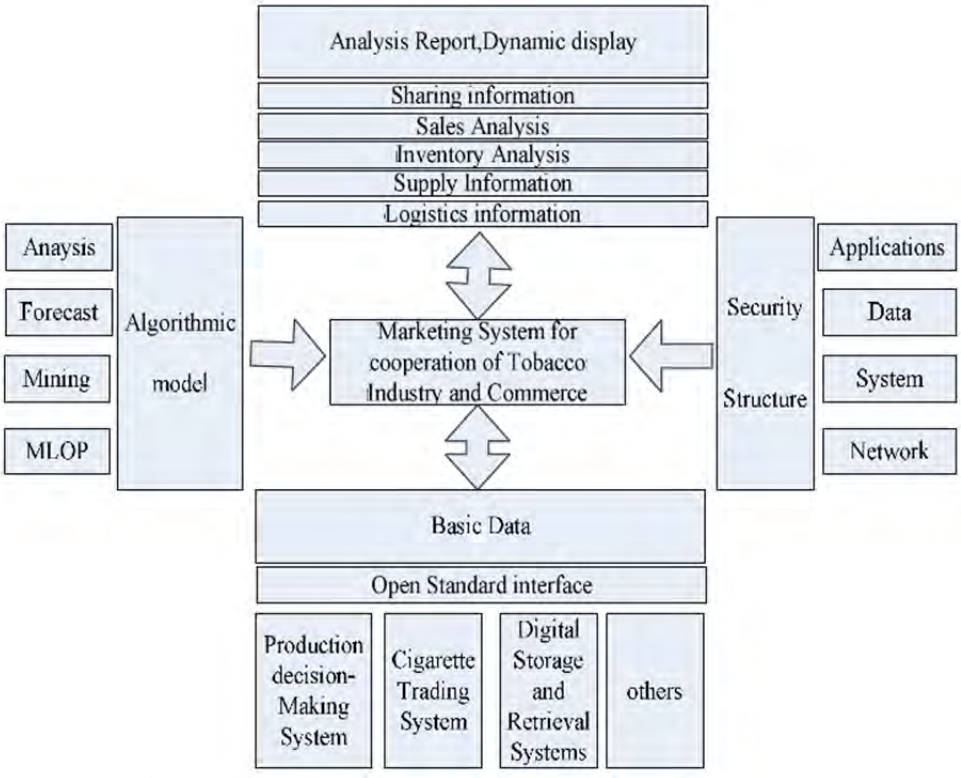


Figure 17. System Architecture based on SaaS pattern (From Bin, Liwei, & Yong, 2009)

The information service system for cooperative marketing of tobacco industry has the following features:

a. Different interface
b. Data structure
c. Business logic for different user

Figure 18 illustrates how the MVC pattern is used in this system.

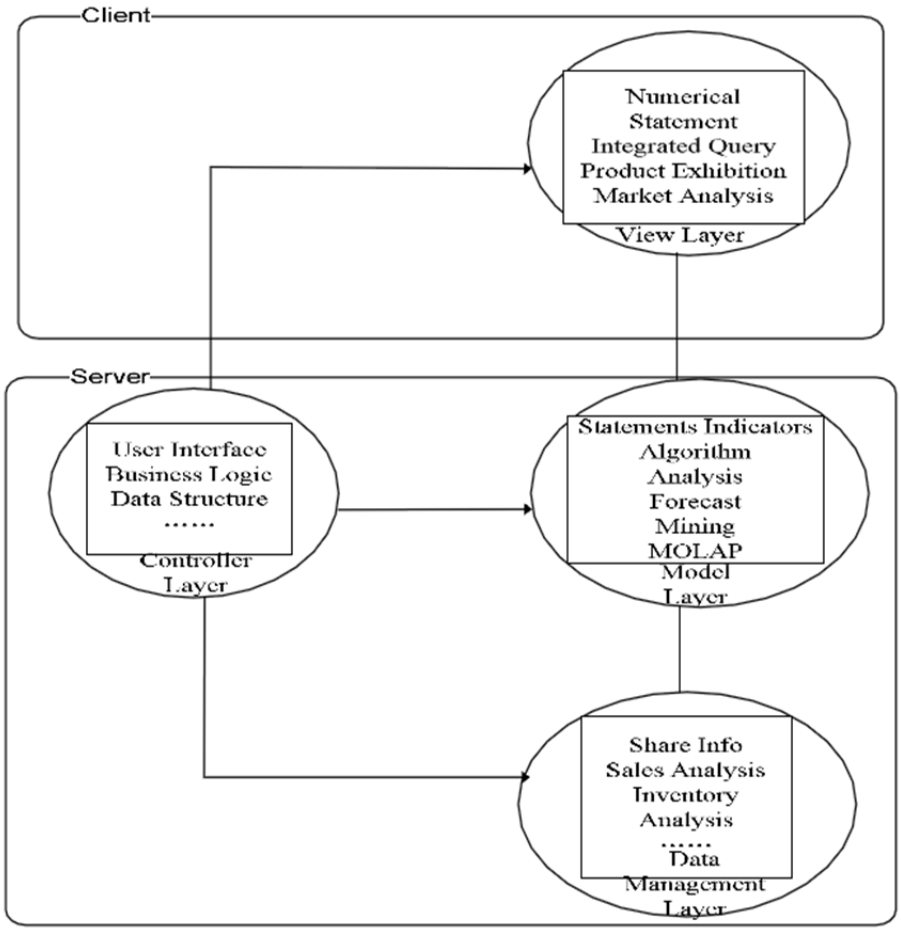


Figure 18. System Design Based the Improved MVC Pattern (From Bin, Liwei, & Yong, 2009)

The implementation of this design provides various business logic operations for multiple users to share, which is achieved by the model. In this system, the model layer, the control layer and the data management layer are all stored and executed on a SaaS server (Bin, Liwei, & Yong, 2009).

3. Operating Auditing System

In the paper, "MVC model based on high-performance computing operating audit system," Zhengquan Zhang and Chengjun Xu presented a job-auditing system that is based on the MVC architecture operating systems and applications Java 2 Platform Enterprise Edition (J2EE) architecture. Using the web, this application gives users the ability to submit job applications; allowing this High Performance Computing Center (HPC) to review the contents of the user's job to guard against malicious code in order to improve the safety and security of HPC (Zhengquan & Chengjun, 2010).

The operating auditing system is based on J2EE three-tier architecture involving JSP, EJB and Servlets, where the Servlet component corresponds to the MVC in the controller part, JSP and Browser corresponds to the view part, while the logic of Bean, and the value of the object correspond in the model part (Zhengquan & Chengjun, 2010). Because the J2EE system is built on the MVC design concept, it encourages the separation of the business logic from the presentation layer.

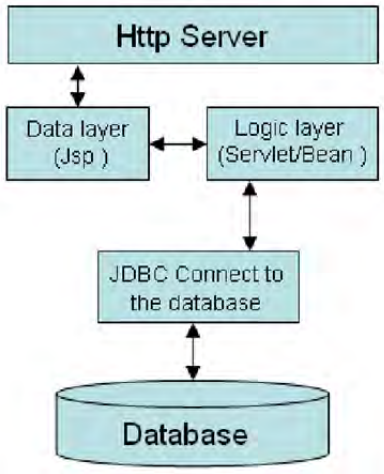


Figure 19. Logical Structure (From Zhengquan & Chengjun, 2010)

The J2EE system is inspired by the MVC design pattern. Zhang and Xu identified the following as the advantages of applying MVC:

a. *"A model can be run at the same time create and use multiple views. Change - dissemination mechanisms to ensure that all relevant changes in*

the view model data in a timely manner so that all the associated view and controller behavior to achieve synchronization."

b. *"View and the controller can Plug-in nature, allowing replacement of the view and controller objects, and can dynamically on demand open or closed, even when running the object during the replacement of."*

c. *"Model portability. Because the model is independent of the view, so a model can be independently migrated to new platforms work. The only thing that needs to be done on the new platform, a new view and controller changes."*

d. *"A potential framework. You can build applications based on this model framework, not only used in the design of interface design."*

4. Desktop Applications with Web Services

In the report, "Building Desktop Applications with Web Services in a Message-based MVC Paradigm," Xiaohong Qiu describes an approach to building desktop application with Web Services in an explicit message-based MVC paradigm. He integrates a publish messaging middleware with a method-based desktop application, Scalable Vector Graphics (SVG) browser (a Microsoft PowerPoint like client application), thus providing desktop applications with Web Service style interfaces and making them universally accessible from different client platforms: Windows, Linux, MacOS, PalmOS and other customized ones (Qiu).

Figure 20 illustrates implementation of MVC architecture. Qiu examines a "universal modular design with messaging linkage service model that converge desktop applications, Web applications, and Internet collaboration to achieve reusability, scalability, interoperability and pervasive accessibility," and proposes an explicit message-based MVC (MMVC) paradigm.

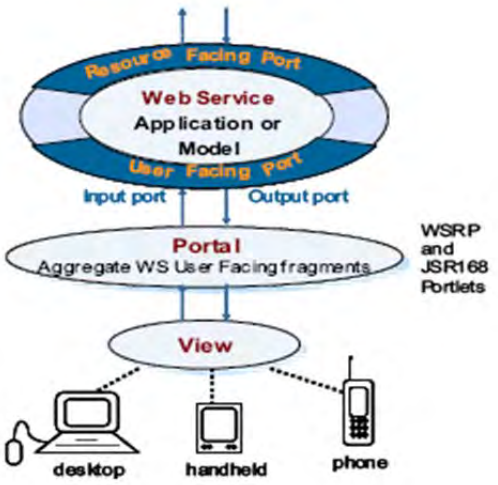


Figure 20. MVC Architecture (From Qiu)

Figure 21 is a general MVC approach. MMVC a different approach with MVC being utilized systematically but with message based interactions, between the model and the view components (Qiu). Qiu believes this approach will give sufficient performance for desktop applications.

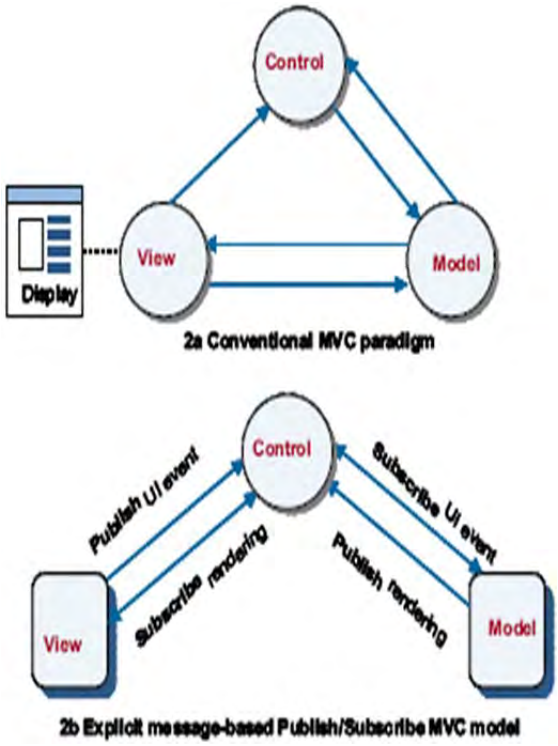


Figure 21. MVC Approach (From Qiu)

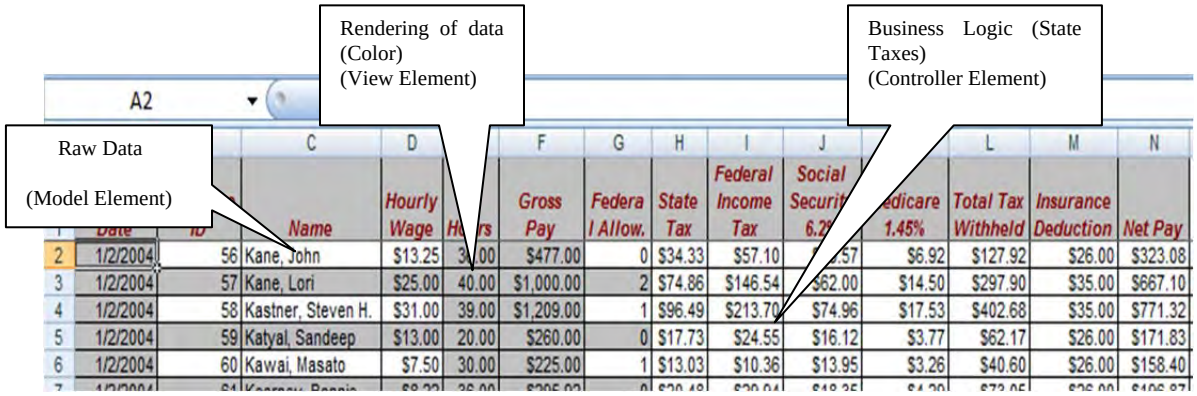
Qiu examines an existing system (Batik SVG browser from Apach). Batik is a Java-based toolkit for applications or applets that want to use images in the SVG format for various purposes, such as display, generation or manipulation (Apache Software Foundation, 2010). Qiu transforms the Batik SVG browser architecture from a method-based desktop application to a message-based one. The message-based architecture permits one to build desktop applications as web services and so join traditional desktop and web service plus portal approaches; thus making collaborative applications straightforward to build (Qiu). The separation of the model and view enables supportive various client devices and operating systems. Qiu's strategy allocates long distance linkage between the *model* and view. Qui uses a similar messaging infrastructure used to support large Grid applications. His approach can be used for interactive applications when model and view are nearby and allow collaboration and traditional web portal use for remote access (Qiu).The prototype exemplified by Qiu demonstrates how a message-based MVC architecture can generate a prevailing application paradigm appropriate for SVG.

THIS PAGE INTENTIONALLY LEFT BLANK

III. APPLYING MVC TO CONTENT MANAGEMENT ON THE CLOUD

Cloud computing provides large organizations the ability to process, store, and share data anytime and anywhere. Content management on the cloud refers to the organization and structure of data on the cloud and the operations of data. Since today's large organizations store, present and share most data in the form of documents, efficient and effective management of document contents will have significant impact on the organizations' ability to use cloud computing to support new workflow and improve their business processes.

Like software, the content of a document are made up of three kinds of elements – model, view and controller. For an example, the model element is the raw data, the view element is the rendering of information (color, positioning and font) and the controller element is business logic (county taxes) (Drusinsky, 2011).



	C	D		F	G	H	I	J	L	M	N
	Name	Hourly Wage	Hours	Gross Pay	Federal Allow.	State Tax	Federal Income Tax	Social Security 6.2%	Medicare 1.45%	Total Tax Withheld	Insurance Deduction	Net Pay
2	56 Kane, John	$13.25	3[illegible].00	$477.00	0	$34.33	$57.10	[illegible].57	$6.92	$127.92	$26.00	$323.08
3	57 Kane, Lori	$25.00	40.00	$1,000.00	2	$74.86	$146.54	$62.00	$14.50	$297.90	$35.00	$667.10
4	58 Kastner, Steven H.	$31.00	39.00	$1,209.00	1	$96.49	$213.70	$74.96	$17.53	$402.68	$35.00	$771.32
5	59 Katyal, Sandeep	$13.00	20.00	$260.00	0	$17.73	$24.55	$16.12	$3.77	$62.17	$26.00	$171.83
6	60 Kawai, Masato	$7.50	30.00	$225.00	1	$13.03	$10.36	$13.95	$3.26	$40.60	$26.00	$158.40

Figure 22. Model, View, and Controller Element in MS Documents

However, all distinct elements of MVC are packaged into a single file in today's documents such as the Word, Excel, and PowerPoint Microsoft Office documents, making them very complex, inflexible, and very difficult to manage. Introducing MVC into document processing, dividing the model, view, and controller elements of a document, will improve the document's reusability, flexibility, and reduce application/platform dependency.

A. BENEFITS FOR APPLYING MVC TO CONTENT IN THE CLOUD

There are many benefits when applying the MVC pattern to manage document content, most of which stem from MVC's enforcement of separation of document elements.

MVC reduces the complexity of the document structure and accommodates change in documents. Improvements to the model, view or controller can be implemented easily. Because MVC enforces separation of elements in a document, users can easily update the model, view, or controller without affecting the whole document. The separation of the document content into smaller modules of model and view also allows finer-grain access control, which facilitate document collaboration by multiple users in the cloud.

MVC promotes reusability. The controller glues together the view and model elements in order to fulfill a request given by the end user. Presented with a number of reusable building blocks in the model and the view, the controller picks and chooses which blocks are needed to handle specific processing and display requirements (Kotek, 2002).

By keeping the view and model elements separated, MVC supports effortless modifications of multiple views of a single document and facilitates the integration of data in different documents (i.e., different views) without the need to physically duplicating them as in today's documents. Moreover, it also enables the generation of complex views via embedded reader-sensitive view logics (e.g., the automatic hiding of sensitive information from unauthorized readers). This separation endorses flexibility.

Incorporating custom business logic in documents allows automated document-level integration and updates of document content, thus improves the accuracy and robustness of the document.

B. USE CASE ANALYSIS

The naval base Facilities Management Department is starting to migrate its documents to the cloud. In order to have better content management, the developers of

these new cloud documents will have to update or redesign the cloud documents before the migration. This section presents two use case analyses to identify the requirements of the cloud documents.

1. Work Request Report Collaboration

This use case analysis focuses on the creation and approval of work request reports. It consists of seven use cases: *Create Work Request*, *Project Manager Evaluation*, *Lead Evaluation*, *Financial Evaluation*, *Estimated Rates*, *Actual Rates*, and *View Facilitates Management Department Document*. In the Work Request report workflow, users must be able to create new work request and the Project Manager, Lead, and Financial must have to ability to evaluate each form and add, edit, and delete rates. The Facilities Management Department document is read-only for ALL employees. The use case diagram below highlights the seven use cases and the actors involved.

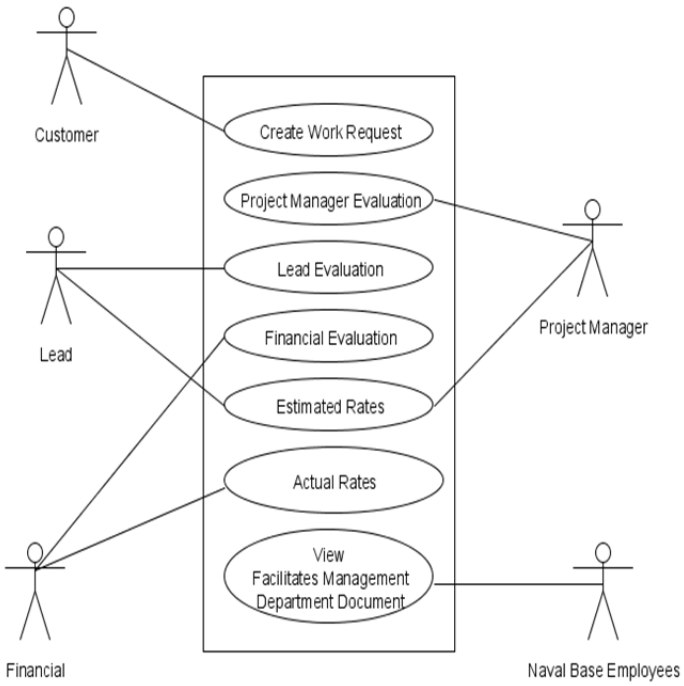


Figure 23. Facilitates Management Department Document Use Case Diagram

a. Actors

The actors below are the entities that are use the reporting system:

a. Customer – This actor initiates the work request.

b. Project Manager – This actor manages the work request and completes the initial estimated funding of the work request.

c. Lead – This actor overseas the work requests to ensure contractors meet the deadline and adequate funding is available for contractors to complete the work.

d. Financial – This actor enters all actual funding and dates at the end of the project.

e. Naval Base Employee – This actor can only view the Facilities Management Department work request documents.

b. Use Cases and Scenarios

a. Create Work Request

   i. Actor: Customer

   ii. Precondition: The customer has a typical engineering or construction project.

   iii. Scenario: The customer opens the work request template, and inputs the following fields: Customer Name, Customer Phone, Customer Code, Customer Email, Priority, Location, Building, Department and Description of Work Request. The customer saves the document to the department's cloud computing document-sharing site.

   iv. Alternative Scenario: NA

   v. Wireframe of Customer Form:

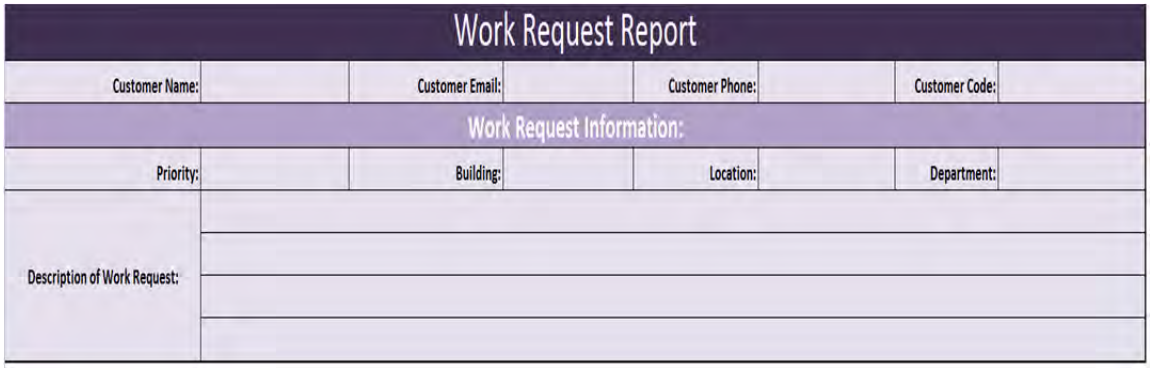


Figure 24. Wireframe of Customer Request Form

b. Project Manager Evaluation

i. Actor: Project Manager

ii. Precondition: A request has been submitted by the customer.

iii. Scenario: The Project Manager (PM) evaluates the work request report to ensure that the data is correct and ensures that the request is an actual engineering or construction project. The manager then enters data into ALL the required fields: Request Work Start, Request Completion Date, Title, Justification, Supplier POC, Supplier Phone, Supplier Fax, PM Name, PM Phone, Approval, Execution Method, Estimation Cost fields. The PM approves the report and saves for review by the Lead.

iv. Alternative Scenario: If the request is disapproved or is not an actual engineering or construction project, the Project Manager removes the work request from the department's site and contacts the customer.

v. Wireframe of PM Form:

Figure 25. Wireframe of PM Form

c. Lead Evaluation

i. Actor: Lead

ii. Precondition: The PM has approved the request and entered data in every cost estimation field.

iii. Scenario: The Lead enters in the appropriate contractors for the requests and ensures that there is adequate funding for contractors to complete the work. The Lead will also enter in the following: Fund Type and Fund Sub-Type.

iv. Alternative Scenario: If there is not enough funding, the Lead will adjust the estimated cost fields.

v. Wireframe of Lead Form:

Work Request Report												
Customer Name:	John.Doe	Customer Email:	john.Doe@navy.mil	PM Name:	Lisa.Dun	PM Email:	Lisa.Dun@navy.mil	Lead Name:		Lead Email:		
Customer Phone:	555-1245	Customer Code:	560A	PM Phone:	555-5687	PM Code:	560A	Lead Phone:		Lead Code:		
				Aprove:								
Contracting Company Information												
Name:	BOSC	Phone:	555-4845	Notes:								
Supplier Information												
Supplier POC:	Sandy Inc	Supplier Pone:	555-1564	Supplier FAX:	555-1815	Notes:	New company					
Work Request Information:												
Priority:	High	Building:	5566	Location:	Navy Base	Department:	560A	Project Type:	Design/Build	Execution Method:	Bankcard	
Funding Type:	Overhead (Dept)	Funding Sub-Type:	Code 50	Notes:								
Title:	Additions for ERP											
Description of Work Request:	Code 560A needs a new conference room, office section, and 5 classrooms before ERP.											
Justification:	ERP											
Request Work Start:	5/10/2011	Request Completion Date:	6/13/2012	Actual Start:		Actual Finish:						
Estimated Costs												
Estimated Material	$500,548	Estimated Equipment	$150,256	Estimated Contract	$100,548	Estimated Design	$125,455	Estimated Labor Hours	380.5			
Labor Rates	$15,220.00	SIOH Rates	20.00%									
estimated total	$892,027.20											
Actual Costs												
Actual Material		Actual Equipment		Actual Contract		Actual Design		Actual Labor Hours				
Labor Rates		SIOH Rates										
Actual total												

Figure 26. Wireframe of Lead Form

d. Financial Evaluation

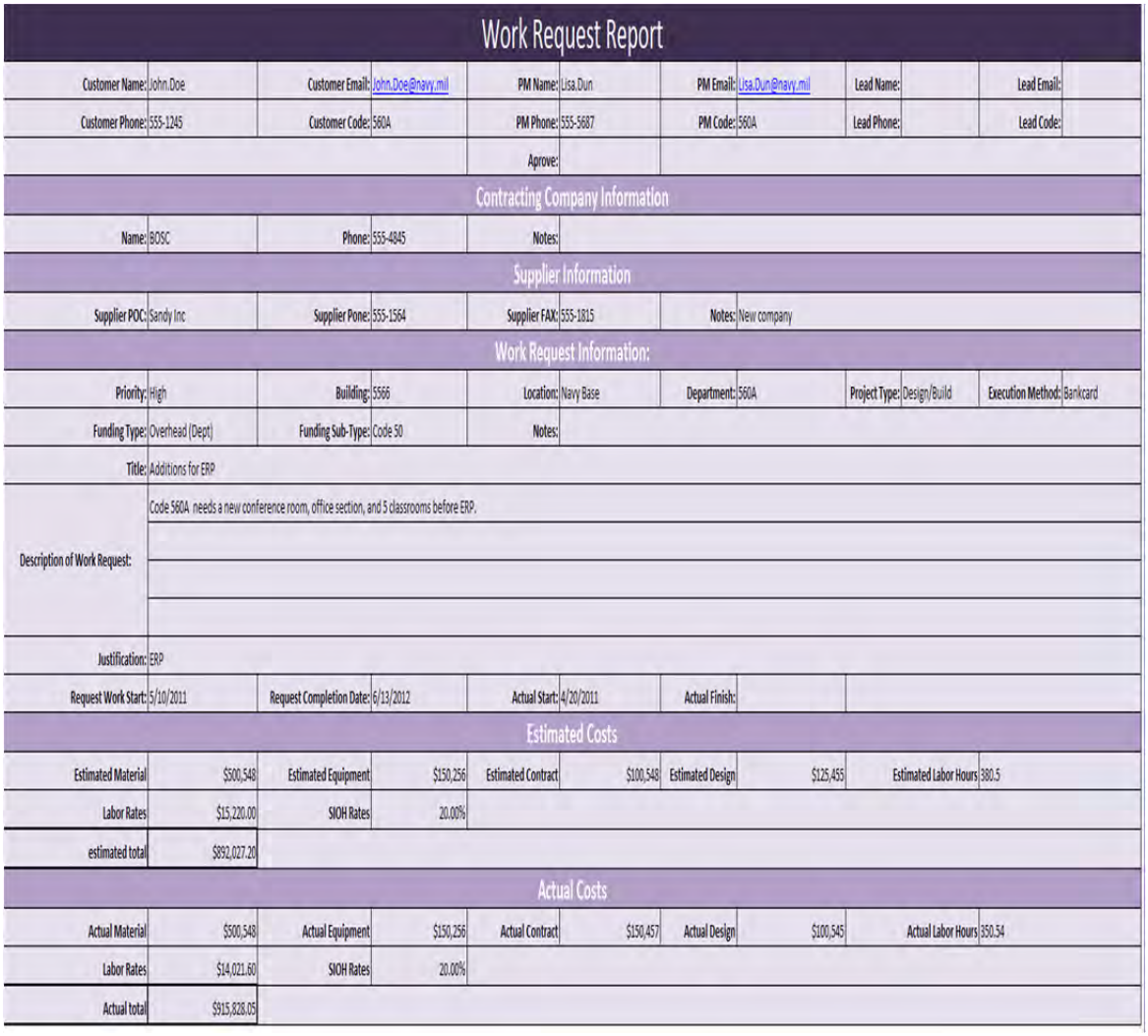
i. Actor: Financial

ii. Precondition: The contractors and funding is entered.

iii. Scenario: The Financial department enters the actual costs.

iv. Alternative Scenario: NA

v. Wireframe of Financial Evaluation Form:



Work Request Report												
Customer Name:	John.Doe	Customer Email:	John.Doe@navy.mil	PM Name:	Lisa.Dun	PM Email:	Lisa.Dun@navy.mil	Lead Name:		Lead Email:		
Customer Phone:	555-1245	Customer Code:	560A	PM Phone:	555-5687	PM Code:	560A	Lead Phone:		Lead Code:		
				Aprove:								
Contracting Company Information												
Name:	BOSC	Phone:	555-4845	Notes:								
Supplier Information												
Supplier POC:	Sandy Inc	Supplier Pone:	555-1564	Supplier FAX:	555-1815	Notes:	New company					
Work Request Information:												
Priority:	High	Building:	5566	Location:	Navy Base	Department:	560A	Project Type:	Design/Build	Execution Method:	Bankcard	
Funding Type:	Overhead (Dept)	Funding Sub-Type:	Code 50	Notes:								
Title:	Additions for ERP											
Description of Work Request:	Code 560A needs a new conference room, office section, and 5 classrooms before ERP.											
Justification:	ERP											
Request Work Start:	5/10/2011	Request Completion Date:	6/13/2012	Actual Start:	4/20/2011	Actual Finish:						
Estimated Costs												
Estimated Material	$500,548	Estimated Equipment	$150,256	Estimated Contract	$100,548	Estimated Design	$125,455	Estimated Labor Hours	380.5			
Labor Rates	$15,220.00	SIOH Rates	20.00%									
estimated total	$892,027.20											
Actual Costs												
Actual Material	$500,548	Actual Equipment	$150,256	Actual Contract	$150,457	Actual Design	$100,545	Actual Labor Hours	350.54			
Labor Rates	$14,021.60	SIOH Rates	20.00%									
Actual total	$915,828.05											

Figure 27. Wireframe of Financial Evaluation Form

e. Estimated Rates

i. Actor: PM, Lead

ii. Precondition: The Execution Method, Fund Type, Fund Sub-Type and Department have been entered in the request form.

iii. Scenario: The PM or Lead enters the estimations for: Labor Hours, Material, Contract, and Design. The Labor Rate and SIOH Rate depend on the Execution Method, Fund Type, and Fund Sub-Type.

The Total Estimation equals to the sum of Labor Rate, Material, Contract, Design, and SIOH rate.

iv. Alternative Scenario: NA

f. Actual Rates

i. Actor: Financial

ii. Precondition: The Execution Method, Fund Type, Fund Sub-Type, and Estimated cost have been entered in the request form.

iii. Scenario: Financial enters the actual numbers for: Labor Hours, Material, Contract, and Design. The Labor Rate and SIOH Rate depend on the Execution Method, Fund Type, and Fund Sub-Type. The Actual Total equals to the sum of Labor Rate, Material, Contract, Design, and SIOH rate.

iv. Alternative Scenario: NA

g. View Facilitates Management Department Document

i. Actor: Naval Base Employees

ii. Precondition: NA

iii. Scenario: The user opens the document from the department's site and views the information.

iv. Alternative Scenario: NA

c. New Requirement for Work Request Document

Recently, a new requirement for the funding rates has been added to the old work request reports. The funding rates are now based on the following new fields: execution method, funding type, and funding sub-type of a project, where the execution method is the method that will be used to execute the payment for the work request such as: Bankcard, PSNS (Puget Sound Naval Shipyard), or SEABEES, and the funding type and funding sub type are the type of funding used in the work request. The user has to

look up the calculation for each execution method, funding type, and funding sub-type field and apply this calculation to the funding rate.

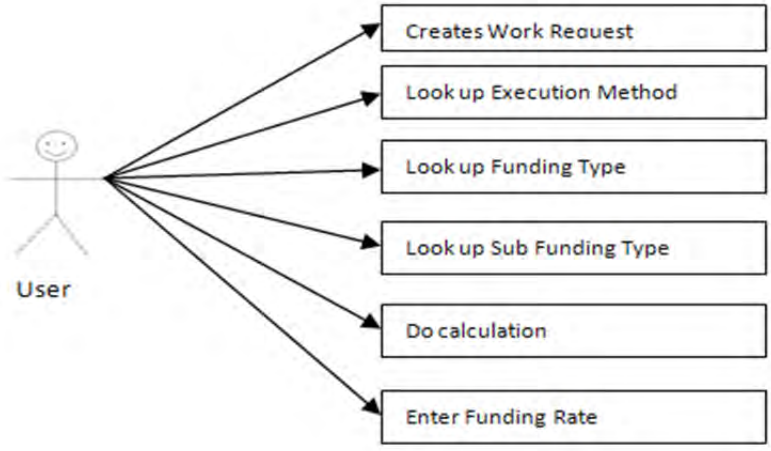


Figure 28. Responsibilities of Work Request User

The new funding rates are deeply rooted in each report and affect six of the seven use cases. All the actors involved need to manually compute and update the funding rate whenever the execution method, funding type, or funding sub-type of a work request changed, thus increasing the likelihood for errors and difficulties in maintaining the content of the document. The brute force approach in computing funding rates necessitates the Facilities Management Department to redesign these reports before migrating them over to the cloud because the contents in this report have become difficult to manage.

Applying the MVC concept to this report will ease the creation and use of these reports, making them more flexible. The MVC will separate the controller (funding rates business logic) from the model (contents) and view (user interface) of the report. The funding rate logic will be stored in an application called FundingAPP.

Work Request Information:												
Priority:	High	Building:	5566	Location:	Navy Base	Department:	560A	Project Type:	Design/Build	Execution Method:	Bankcard	
Funding Type:	Overhead (Dept)	Funding Sub-Type:	Code 50	Notes:								
Title:	Additions for ERP											
Description of Work Request:	Code 560A needs a new conference room, office section, and 5 classrooms before ERP.											
Justification:	ERP											
Request Work Start:	5/10/2011	Request Completion Date:	6/13/2012	Actual Start:	4/20/2011	Actual Finish:						
Estimated Costs												
Estimated Material	$500,548	Estimated Equipment	$150,256	Estimated Contract	$100,548	Estimated Design	$125,455	Estimated Labor Hours	380.5			
Labor Rates	$15,220.00	SIOH Rates	20.00%									
estimated total	$892,027.20											
Actual Costs												
Actual Material	$500,548	Actual Equipment	$150,256	Actual Contract	$150,457	Actual Design	$100,545	Actual Labor Hours	350.54			
Labor Rates	$14,021.60	SIOH Rates	20.00%									
Actual total	$915,828.05											

Figure 29. FundingApp Fields

Now, when a user creates a work request, he/she no longer has to worry about calculating the funding rate because the FundingApp will automatically calculate the rate depending on the execution method, funding type, and funding sub-type fields.

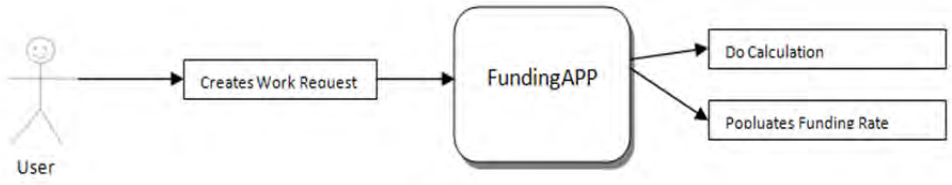


Figure 30. New Responisibilties of Work Request User

In Figure 30, notice the new responsibilities of the user when creating work request. The user is only responsible for creating the work request report and the FundingApp performs the logic and populates the funding rate field.

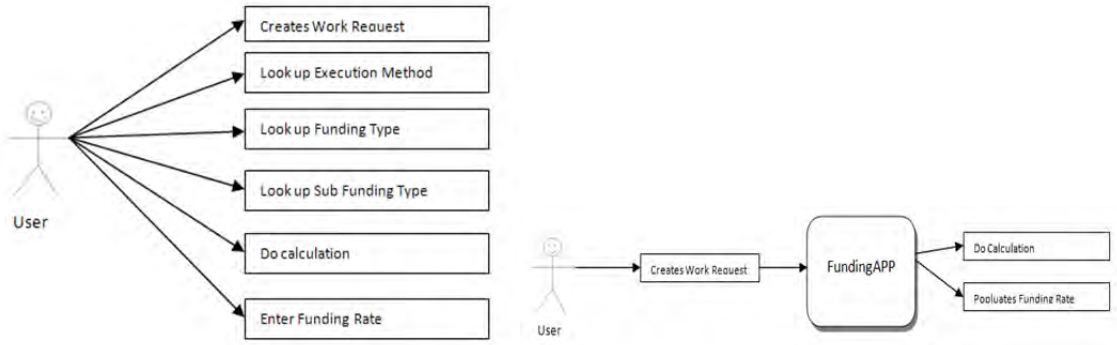


Figure 31. Comparsion of Old and New Document

In the old document without the MVC approach, the user has more steps in completing a work request report. In the new document with the MVC approach, the user only has one step in completing a work request report because the business logic will be done in the FundingApp. The above illustration only demonstrates the creation of a work request by one user either the Project Manager, Lead, or Financial; however; the use case analysis and requirements will divide a user into four groups: Customer, Project Manager, Lead, and Financial. The use case analysis and requirements will provide more detail about the different roles and responsibilities of each user and the different fields in the work request form.

2. Update of the Organization Logo

The Facilities Management Department has also recently changed the logo in a 500 page Facilities Management Department document. In this document, the logos have changed thus requiring the views of the document to change. Logos are throughout the document; for instances, logos can be found in headers, footers, paragraphs, etc. Usually, when updating logos/images the user will have to manually update each page containing the out-dated logos/images to the new logos/images. Applying MVC concept to this document can allow these logos to be easily changed using Open Office XML (OOXML). When using OOXML to update the logos, the user only has to change the logo in one place and Microsoft Word will do the rest if the work. OOXML separates contents of a document into different folders. The media folder holds the images of a document. The user will use this folder to replace ALL of the logos in this document. The

only concern is the view of the document therefore the model and controller will stay the same. When a document uses MVC, a single element can be updated or changed without manipulating the other to elements.

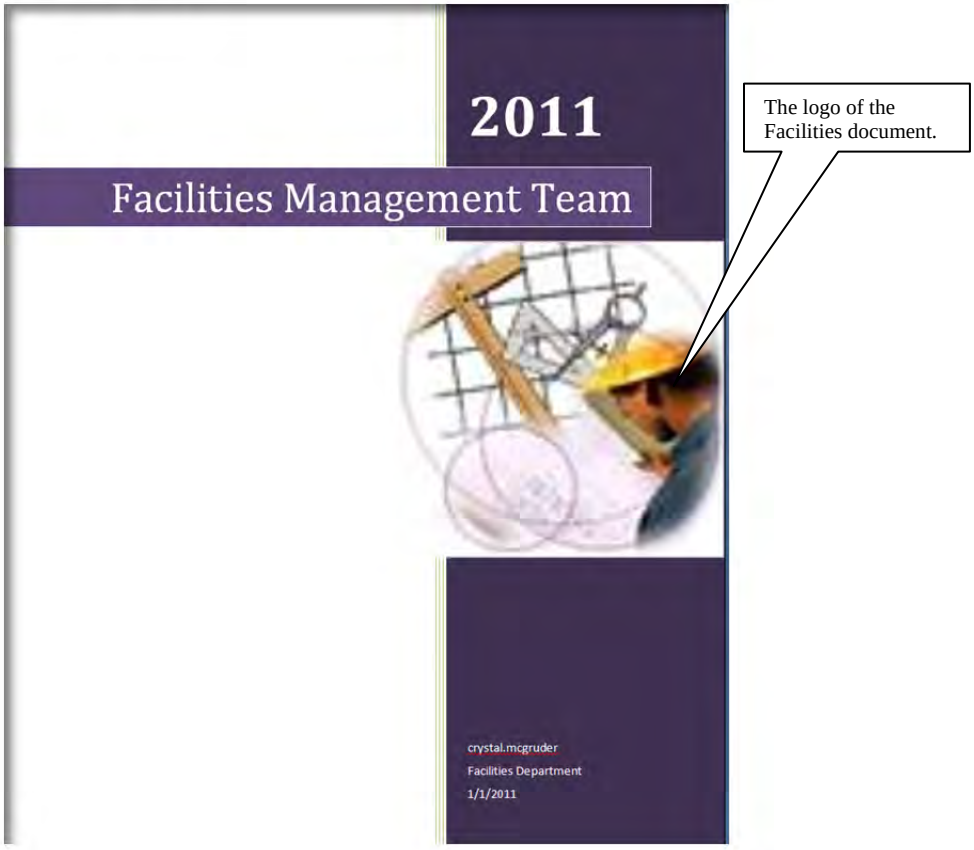


Figure 32. The View Element of the 500 Page Report

C. REQUIREMENTS OF THE WORK REQUEST DOCUMENTS

Below lists all of the requirements and constraints of the Work Request cloud documents.

1. Functional Requirements

a. A work request is a typical engineering or construction project created by the customer.

b. A work request MUST be approved by a Project Manager.

c. The estimations funding fields are: Estimated Material, Estimated Equipment, Estimated Contract, Estimated Design, Estimated Labor Hours, Labor Rates, and SIOH Rates.

d. The actual funding fields are: Actual Material, Actual Equipment, Actual Contract, Actual Design, Actual Labor Hours, Labor Rates, and SIOH Rates.

e. The estimated total is the sum of Estimated Material, Estimated Equipment, Estimated Contract, Estimated Design, Labor Rates, and SIOH Rates.

f. The actual total is the sum of Actual Material, Actual Equipment, Actual Contract, Actual Design, Labor Rates, and SIOH Rates.

g. The execution method should ONLY include the following data: ACOE, Bankcard, DLA-SAIC, EMALL, GSA, HQ ASC, NAVFAC SCAN, NAVSEA-EB, NFEC HI, NFECSW, PSNS, SEABEES and Other

h. The funding type should ONLY include the following data: TBD, BRAC, CIP (17), CIP (Dept), Direct, FSCC, MILCON, Overhead (17), Overhead (Dept), USCC and Other.

i. The funding sub-type should include only the following data: Code 10, Code 20, Code 30, Code 40, Code 50, Code 60, Code 70, Other

j. The contracting companies should include: INDUS, BOSC, RNISH, TRANE, and NBK Alarm Shop

k. The project type should include only the following data: Design, Design/Build, Equipment, Study, Construction, Alterations

l. The calculations for work requests are the following:

    i. If execution method is ACOE and funding type is equal to BRAC, CIP (17), CIP (Dept), or Direct and funding sub-type is equal to Code 10, Code 20, Code 30, Code 40, Code 50, Code 60, or Code 70 then SiOH Rate is %2.25

1. Or, if execution method is DLA-SAIC and funding type is equal to TBD, BRAC, CIP (17), CIP (Dept), Direct, FSCC, MILCON, Overhead (17), Overhead (Dept), USCC or Other and funding sub-type is equal to Code 10, Code 20, Code 30, Code 40, Code 50, Code 60, or Code 70 then SiOH Rate is %2.25
2. Or, if execution method is DLA-SAIC and funding type is equal to BRAC, CIP (17), CIP (Dept), or Direct and funding sub-type is equal to Code 10, Code 20, Code 30, Code 40, Code 50, Code 60, or Code 70 then SiOH Rate is %5.45
3. Or, if execution method is NAVSEA-EB and funding type is equal to BRAC, CIP (17), CIP (Dept), or Direct and funding sub-type is equal to Other then SiOH Rate is %8.30
4. Or, if execution method is PSNS and funding type is equal to BRAC, CIP (17), CIP (Dept), or Direct and funding sub-type is equal to Code 10, Code 20, Code 30, Code 40, Code 50, Code 60, or Code 70 then SiOH Rate is %12.85
5. Or, if execution method is PSNS and funding type is equal to other and funding sub-type is equal to other then SiOH Rate is %.054; else SiOH Rate is %1.

m. The work request template should include the following fields: Customer Name, Customer Phone, Customer Code, POC Name, POC Phone, POC Code, Priority, Location, Building, Department, Description of Work Request, Request Work Start, Request Completion Date, Title of Project, Justification, Supplier POC, Supplier Phone, Supplier Fax, Project Manager Name, Project

Manager Phone, Execution Method, Contracting Company, Project Sub Type

n. A business logic application called FundingApp will automatically calculate the rates for the user based on the execution method, funding type and funding sub-type.

2. Non-Functional Requirements

a. The Facilities Management Department must have a cloud computing document-sharing site.

b. Replace the old logo in the Facilitates Management Department document with the new logo.

c. Separate the rates business logic from the data in the Work Request report and place into a business application called FundingApp.

d. Documents are created in Microsoft Office 2007 or higher version.

D. CLOUD DOCUMENTS CLASS DIAGRAM

This section presents a class diagram and a package diagram for the Work Request document. These diagrams will describe the structure of the documents by showing the different objects and their relationship.

1. Work Request Class Diagram

The class diagram below shows that the Facilities Management Department uses many documents, where a document can have one or many logos. It also shows the concepts involved in the creation and evaluation of a work request, which is a kind of Facilities Management Department documents. The class diagram also highlights the responsibilities of the project manager, lead and financial.

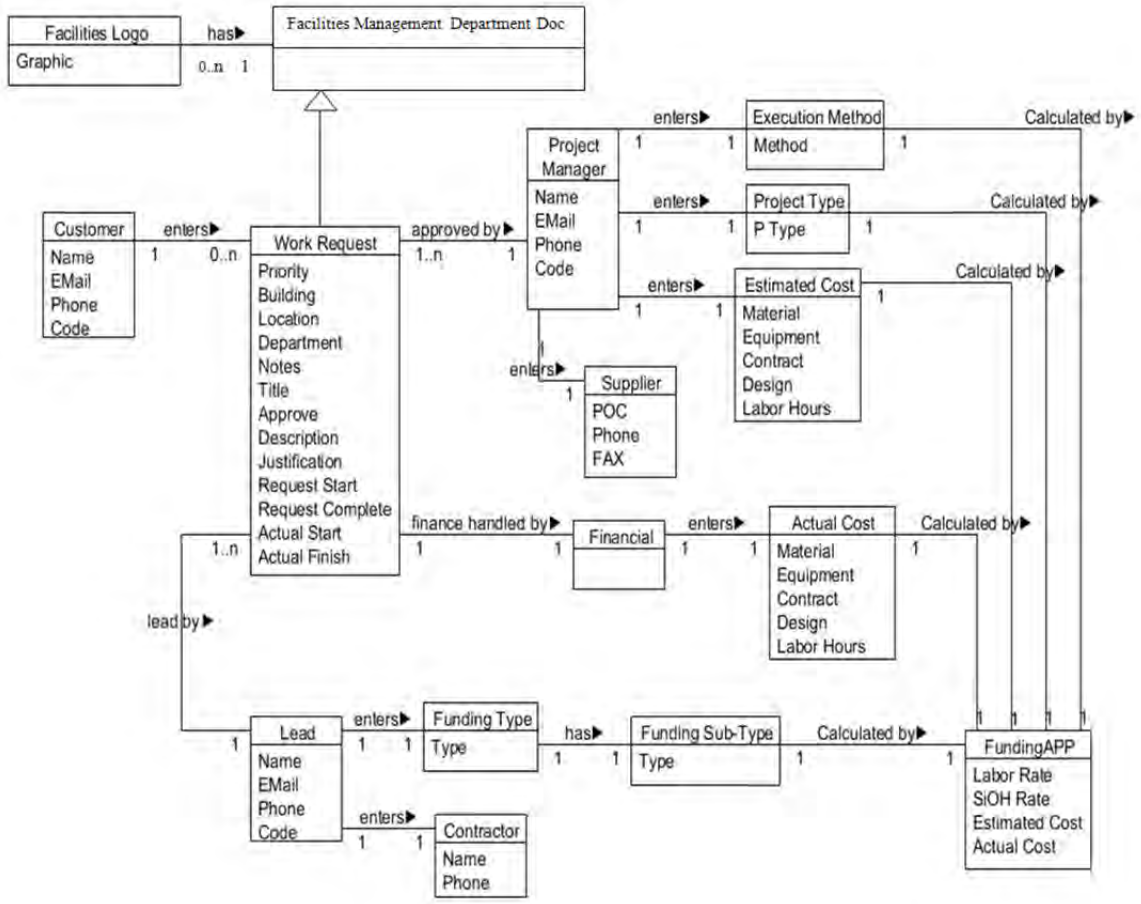


Figure 33. Facilities Management Department Documents Class Diagram

2. Work Request Package Diagram

The work request package diagram illustrates the separation of the work request report into the MVC three elements: model, view and controller. The view package includes the logo graphics and the work request report. The model package is the raw data in the work request report. The controller package is the FundingApp of the work request report that performs all calculations.

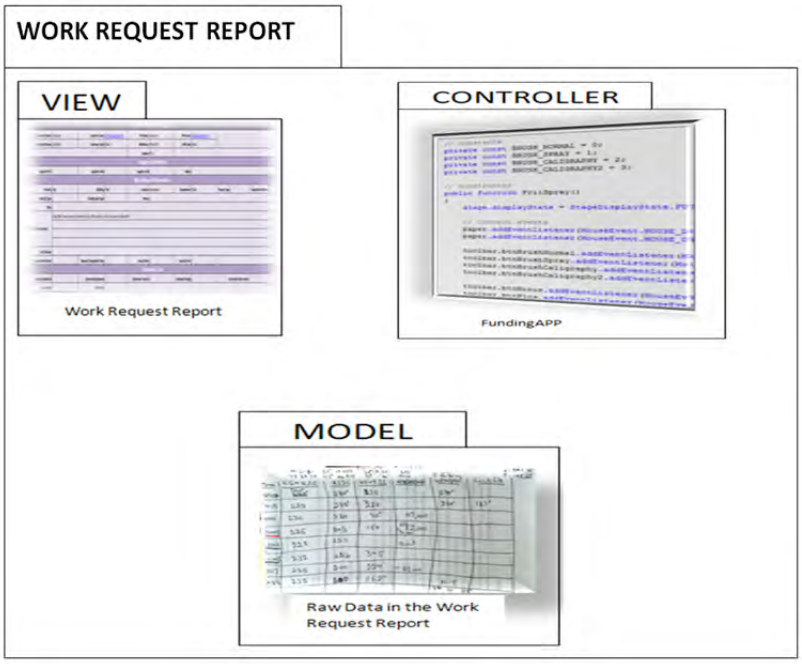


Figure 34. Work Request Report Package Diagram

IV. DOCUMENT STANDARDS IN SUPPORT OF MVC

In this chapter, we review three existing document standards and assess their readiness in supporting the MVC document architecture.

A. EXISITNG OPEN DOCUMENT STANDARDS

1. Office Open XML

Microsoft Word, Excel, and PowerPoint 2007 have new file formats. These file formats will reduce file size, improve security and reliability, and enhance integration with external sources (Microsoft Corporation, 2011). Office Open XML (OOXML) is the default target file format of Microsoft Office. OOXML uses an open standard, thereby making the file formats accessible through a plurality of applications in which is seemingly in concert with MVC pattern (Drusinsky, 2011).

According to Wikipedia, OOXML is a zipped XML-based file format for representing spreadsheets, charts, presentations and word processing documents. OOXML was standardized by Ecma (Wikipedia MVC). It is the new file format for word-processing, presentations, and spreadsheet documents. The primary markup language for each document is: WordprocessingML, PresentationML, and SpreadsheetML. There are other markups in OOXML such as DrawingML and Custom XML.

The Markup Compatibility is also a very important specification because it has the capability for a document expressed in WordprocessingML, SpreadsheetML, or PresentationML markup languages to facilitate interoperability between applications. The diagram below is an overview of the layers of specification.

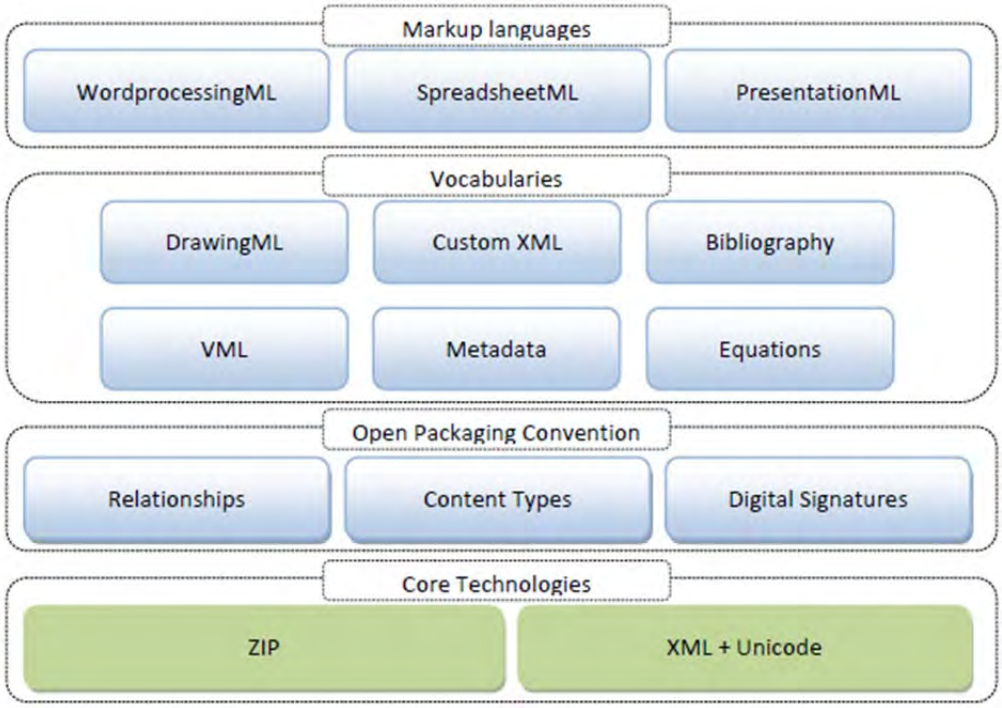


Figure 35. Components of OOXML (From Vugt, 2007)

An OOXML document is a container with many parts. The container is typically implemented as a ZIP file, and the parts can be viewed as files within the ZIP. Moreover, one could also store the document parts in a database to maximize reuse (Vugt, 2007). The structure inside the container is the Open Packing Convention (OPC). The OPC provides a way to store multiple type of content (e.g., XML, images, and metadata) in a container, such as a ZIP archive, to fully represent a document

The collection of components that comprise the document within in a ZIP file is referred to as a package. Parts and relationship items are the two types of components. A part is the same as a file in a package. Relationships are stored inside the parts of a package. A package is implemented as a ZIP archive, with each component in a package corresponding to an item in the archive (Madhva). Component pieces such as images, fonts, and data are circulated to a document through a package. A package merges all the pieces of a document into a single file.

2. ODF

The Open Document Format Office application known as OpenDocument (ODF) is an XML-based free open format for office documents such as spreadsheets, charts, presentations and word-processing documents. Sun Microsystems developed the specifications for ODF. The Organization for the Advancement of Structured Information Standards (OASIS) standardized ODF. The most common ODF document filename extensions are: .odt for word processing (text) documents, .ods for spreadsheets, .odp for presentations, .odb databases, .odg for graphics, and .odf for formulae.

ODF offers the following: Long-term reuse of and access to data, no lock-in to proprietary tools or undocumented formats, competitive data processing products, reduced costs, increased reliability, platform independence and interoperability (Edstorm, 2005). The ODF is an idealized representation of a document's structure allowing any applications using ODF to implement new features or completely alter internal data structures without requiring major changes to the file format (O'Reilly & Associates, 2005). Java Archive (JAR) format is used to store ODF files; it is a compressed ZIP file with additional "manifest" file that has a list of contents of the archive.

Figure 36 shows a word processing document saved as ODF document "firstdoc.odt."

Figure 36. Text document (From O'Reilly & Associates, 2005)

```
[david@penguin ch01]$ unzip -v firstdoc.odt
Archive:  firstdoc.odt
 Length   Method    Size  Ratio Name
--------  ------  ------- -----  ----
      39  Stored       39   0%  mimetype
    3441  Defl:N      885  74%  content.xml
    6748  Defl:N     1543  77%  styles.xml
    1173  Stored     1173   0%  meta.xml
     642  Defl:N      345  46%  Thumbnails/thumbnail.png
    7176  Defl:N     1307  82%  settings.xml
    1074  Defl:N      308  71%  META-INF/manifest.xml
       0  Stored        0   0%  Configurations2/
       0  Stored        0   0%  Pictures/
--------          -------  ---  -------
   20293             5600  72%  9 files
```

Figure 37. Listing of Unzipped Text Document (From O'Reilly & Associates, Inc, 2005)

Figure 37 shows the results of unzipping this file in Linux. The contents of the files are as follows:

a. Mimetype: single line of text that provides MIME type for the text document

b. Content.xml: content of the text document

c. Styles.xml: data about the content's style to separate content from presentation

d. Meta.xml: contains data such as the author, late revision, date, etc.

e. Settings.xml: contains data specific to the application

f. META-INF/manifest.xml: contains a list of other files in the JAR

g. Configurations2: contains configuration data

h. Pictures: contains all the images from the text document

ODF supports MVC because when documents are saved as ODF documents, contents of the documents are saved into multiple flat files that allow users to edit/update different files without affecting other files.

3. Rich Text Format

Rich Text Format (RTF) is a document standard that enables encoding many different text formatting properties, such as bold characters and typefaces, as well as document format and structures (Indiana University, 2011). This method can be used to transfer formatted text and graphics between applications. If you save a document in RTF, it can be open in many word processors and other RTF-aware software packaging while keeping of the formatting unchanged. RTF mostly involves the view element of MVC. RFT allows portability of the view of a document. RTF uses the American National Standards Institute (ANSI), PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print (Microsoft MSDN, 2011).

The RTF writer is used by software to convert a formatted file to a RTF file. The RTF reader is included when software converts a RTF file into a formatted file. The RTF writer separates the application's control information from the actual text and writes a new file containing the text and the RTF groups associated with that text (Microsoft MSDN, 2011).

RFT is supported by Microsoft Office. RTF is used throughout the Microsoft Office System to perform functions such as opening documents into Word and Cutting and Pasting content (Brown, 2009).

B. EVALUATION OF EXISTING STANDARDS

In this section, each existing standard identified above (OOXML, ODF and RTF) is evaluated to access its readiness to support the MVC requirements.

1. Evaluation of OOXML

The new Microsoft Office Open XML Formats combine the power of the world's most widely used productivity programs with the integration capabilities enabled by XML (Microsoft Corporation, 2005). In OOXML format files, data is stored

independently which allows modularity. This file format also present improved reliability and notably reduce file sizes. On the MSDN and Microsoft websites, the benefits of using OOXML are identified as:

a. Compact files: Files are instantly compressed. This is done through the use of zip-compression technology to store documents which will reduces the disk space required to store files, and decreases the bandwidth needed to send files via e-mail, over networks, and across the Internet (Microsoft Corporation, 2011)

b. Improved damaged-file recovery: If a component in a file has been damaged, OOXML will keep different data components in the file separate from each other.

c. Better privacy and more control over personal information: "Documents can be shared confidentially, because personally-identifiable and business-sensitive information, such as author names, comments, tracked changes, and file paths can be identified and removed by using Document Inspector."

d. Better integration and interoperability of business data: OOXML allows anyone to use documents and license, royalty free. Customer-defined XML schemas are supported to improve the existing Office document types which will allow customers to unlock data in the current system and use it in acquainted Office programs. The ability to save, load, and use the document format in a wide variety of applications and tools means that we have document interoperability in a way we have never seen before (White, 2009).

e. Easier detection of documents that contain macros: Visual Basic for Applications and XML macros allowed in files that are saved using the "x" suffix.

f. Backwards Compatibility: The ability to convert binary documents to Open XML with a high degree of fidelity means that companies who have literally millions of documents can convert them, and 'light up' these existing documents (White, 2009).

g. Programmability: To use OOXML all you need is a library allowing zip files to be open and saved plus a XML parser/processor.

2. Evaluation of ODF

ODF is essential because it offers cost efficiency and manages the use of documents. Tools that support ODF allow users to use many different vendors for their content because ODF was designed to be vendor neutral. ODF is the only official standard that meets the need of the government. For instance, the government's

documents must be available for many decades, and available to any citizen despite of the equipment they utilize; ODF will fulfill this need.

ODF can be used by many different applications. Established standards (XHTML, SVG, XSL, SMIL, XLink, XForms, MathML, and Dublin Core) are reused in ODF to simplify transformations and increase interoperability. ODF files of different application types (e.g., the word processor, spreadsheet) include the same set of XML files within the ZIP packages (OASIS ODF Adoption TC, 2006).

In Table 1, the "OASIS ODF Adoption TC" team identifies the key features and benefits of ODF.

Table 1. Key Features and Benefits of ODF (From OASIS ODF Adoption TC, 2006)

Feature	Benefit
OASIS standard	Open, transparent specification process with multi-vendor participation
Approved by ISO as ISO/IEC 26300	Well known and broadly accepted standard
ISO standard Relax-NG schema types (ISO/IEC19757-2:2003)	Well known and broadly accepted standard
Supported by multiple applications	Choice between free, open-source and commercial implementations including OpenOffice.org, StarOffice, KOffice, IBM Workplace, Textmaker, Abiword/Gnumeric, Google Docs & Spreadsheet, and AjaxWrite.
Broad industry support	ODF guarantees long-term viability. The OASIS ODF TC, the OASIS ODF Adoption TC, and the ODF Alliance include members from Adobe, BBC, Bristol City Council, Bull, Corel, EDS, EMC, GNOME, IBM, Intel, KDE, MySQL AB, Novell, Oracle, Red Hat, Software AG, Sun Microsystems, and the City of Vienna. As of December 2006, the ODF Alliance already has more than 350 members.
Shipping products since September 2005	ODF files can already be created and used today. The first products with ODF support started

	shipping in September 2005.
Free open source “reference” implementations	ODF is supported by multiple free, opensource office applications including OpenOffice.org, KOffice and Abiword/Gnumeric. OpenOffice.org, for example, is developed by a large community including vendors like Sun Microsystems, Novell, Intel, and Red Hat. Because the source code is available, anyone can add support for additional platforms.
ODF implementations available for all major desktop platforms	Applications with ODF support are available for Microsoft Windows, Linux, the Solaris OS, Apple Mac OS X, and FreeBSD.
Open standard W3C XForms technology is used for forms	The forms concept integrated into ODF is based on the W3C XForms standard which is supported by multiple applications and vendors.
Reuse of existing standards where possible	In order to make interoperability as simple as possible, ODF reuses established standards such as XHTML, SVG, XSL, SMIL, XLink, XForms, MathML, and Dublin Core.
Well established	The first work for the ODF file format started as early as 1999.

3. Evaluation of RTF

RTF makes the exchanging of documents among different word processors and operating systems simple. For example, one can send an RTF file created in Microsoft Word 2002 using Windows XP to someone who uses Word 97 in Windows 98, WordPerfect 6.0 on Windows 3.1, StarOffice on a Linux system, or Word 5 on a Macintosh system and all users will be able to open and read the file regardless of the software version or operating system (Department of Translation Studies, University of Tampere, 2011). RTF includes the entire text markup that is included in the original document enabling it to print, and look, like the original document.

RTF is very easy to use. It can be used in Microsoft Word, Corel WordPerfect and other word processing programs. To use RTF, just choose the RTF when saving. There are many reasons to use RTF: Widely Readable, Preserves Basic Formatting, Security Advantages, Smaller File Size.

RTF can be read by majority of word processors and other type of related programs. RTF can also be used by desktop database applications and email clients. It works in any operating system.

RTF saves the font selection, font sizing, text styling and font coloring. It is often used in the publishing world as a format for rough drafts (Senior Corps Tech Center, 2008). For instance, once editing of a document is completed, the RFT is imported into a page layout program for final formatting which allows editors to focus on editing content without the distraction of too much formatting (Senior Corps Tech Center, 2008).

RTF is a trustworthy format because it cannot contain any macros. Macros can be used to transport viruses. One advantage that RTF files have over several newer word processing formats is its inability to support macros which are a set of automated instructions thus preventing malware being embedded in the file (Hogan, 2011). On the other hand, RTF allows anyone to read data in the file making it not secure.

A RTF file does not require much memory because it does not contain numerous macros or complex formatting information thus allowing uploading, downloading, and e-mail transmission times to be quicker. Conversely, if the document contains embedded graphics, audio files, etc., then an RTF version may be considerably larger than the DOC version, as RTF would have to convert highly-complex graphics information into RTF format, which would involve a considerable amount of coding instruction (Department of Translation Studies, University of Tampere, 2011).

C. SUMMARY OF FINDINGS

The MVC design pattern is a well-established and compelling approach for document processing. Documents are separated into three elements. True MVC of document processing involves separation. ODF and OOXML are more in concert with MVC than RTF; and they both can be used to apply the MVC concept. Even though RTF provides flexibility and portability, it only pertains to the formatting of a document (view element of MVC). OOXML allows the file formats to be accessible through many of applications. ODF is vendor neutral; customers can access ODF regardless of vendor. Both ODF and OOXML also enforce separation.

V. CONCLUSION

A. THESIS SUMMARY

This thesis presents the MVC architecture, describes its elements and explains and how the elements work together. It then presents a use case analysis to capture the requirements in the cloud documents collaboration, control and management in the cloud.

We presented a use case study to illustrate how MVC can be applied to cloud documents. The study applies MVC by separating the controller of the document (business logic) from the view and model of the document. This separation will give the Facilities Management Department the ability to control and manage the content of the document without affecting other elements of the document. We also presented another use case to illustrate how OOXML can be used to support MVC by separating the view of the document (images/logos) from the controller and model of the document allowing all the images/logos of the document to be changed without affecting the controller and the model of the document.

B. FUTURE WORK

To improve reusability and flexibility in processing and managing document contents, the MVC design pattern should be used to divide the document structure and processing into model, view and controller. MVC can be readily supported by document standards such as OOXML and ODF. Although existing document processing software already provides user capability to include business logic (in the form of macros and hyperlinks), new MVC-aware software is needed to provide user with fine-grained version and access control to the MVC elements. We need to review other document standards to assess where they stand in their support for MVC. In particular, we need to examine XForms, a new document technology that is based on the MVC approach. We also need to do more research on new ways to embed business logic in documents and how we can use them for the controller element.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Apache Software Foundation. (2010). *Batik SVG toolkit*. Retrieved July 27, 2011, from http://xmlgraphics.apache.org/batik/

Battlez. (2007, October 06). *Model view controller*. Retrieved March 22, 2011, from http://www.phpwact.org/pattern/model_view_controller

Bin, W., Liwei, B., & Yong, Y. (2009). *Study on the Information Service System of Cooperative Marketing of Tobacco Industry Based on SaaS.* Hefei, China: International Forum on Information Technology and Applications.

Brown, M. (2009, March 19). *The Workshare Blog Benefits of the RTF Document Format*. Retrieved July 15, 2011, from Workshare: http://www.workshare.com/community/blogs/workshare/archive/2009/03/19/benefits-of-the-rtf-document-format.aspx

Cochran, M. (2005, December 12). *Introduction to Model View Control (MVC) Pattern using C#*. Retrieved March 22, 2011, from http://www.c-sharpcorner.com/uploadfile/rmcochran/mvc_intro12122005162329pm/mvc_intro.aspx?articleid=448db537-f236-497d-a16b-46c5d1141e3f remove all

Department of Translation Studies, University of Tampere. (2011, March 03). *Why Should One Use RTF (Rich Text Format) Files?* Retrieved July 2011, from Digital Literacy and Academic Knowledge Management: http://www.uta.fi/FAST/PK5/rtf.html

Ditch, W. (2007). *XML-based office document standards.* Bristol, UK: JISC

Drusinsky, D. (2011). *MVC-based Content Management on the Cloud* (technical report). Naval Postgraduate School.

Edstorm, D. (2005). Retrieved July 15, 2011, from Sun Microsoft System: http://xml.gov/presentations/sun/odf.pdf

Fowler, M. (2002). Transform View. In M. Fowler, *Patterns of enterprise application architecture* (p. 361). Addison-Wesley Professional.

Fowler, M. (2006, July 18). *GUI Architectures*. Retrieved March 2011, 2011, from http://martinfowler.com/eaaDev/uiArchs.html

Gérardin, O. (2009, March 18). *Why Ext-GWT MVC is broken*. Retrieved March 18, 2011, from http://blog.gerardin.info/archives/40

Hogan, B. (2011). *What is RTF?* Retrieved July 2011, from eHow: http://www.ehow.com/facts_5591951_rtf_.html

Indiana University. (2011, January 27). *Knowledge Base*. Retrieved July 15, 2011, from University Information Technology Services: http://kb.iu.edu/data/adnl.html

Joomla. (2010). *Create MVC-model-view-controller component for joomla 1.5 - Hello World*. Retrieved March 22, 2011, from http://www.vojtechovsky.net/joomla/component-helloworld-2-create-tutorial-guide-en.html

Kelly, S., & Mazyck, C. (2010). *Cloud Computing in Support of Synchronized Disaster Response Operations* (master's thesis). Naval Postgraduate School.

Kotek, B. (2002, 10 30). *MVC design pattern brings about better organization and code reuse*. Retrieved June 10, 11, from TechRepublic: http://www.techrepublic.com/article/mvc-design-pattern-brings-about-better-organization-and-code-reuse/1049862

Madhva, S. K. (n.d.). *Creating word document in office open xml format using jave*: openxmldeveloper.org

Mahugh. (2007). *Open XML the markup explained.* Redmond, WA:Microsoft.

Mell, P., & Tim, G. (2009, 10 07). *The NSIT Definition of Cloud Computing.* Retrieved May 24, 2011, from www.au.af.mil/au/awc/awcgate/nist/cloud-def-v15.doc

Microsoft Corporation. (2005, June). Retrieved July 2011, from Microsoft: (windowsconnected.com/media/p/2899/download.aspx).

Microsoft Corporation. (2011). *Introduction to new file-name extensions*. Retrieved July 2011, from Microsoft Office: http://office.microsoft.com/en-us/help/introduction-to-new-file-name-extensions-HA010006935.aspx

Microsoft MSDN. (2011). *Introduction*. Retrieved July 15, 2011, from MSDN: http://msdn.microsoft.com/en-us/library/aa140280(v=office.10).aspx

Model view controller (n.d.). Retrieved March 22, 2011, from a Wikipedia website on MVC: http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller update any in-text citation: (search term, n.d.). re-alpha this endtry.

MSDN. (n.d.). *Model-View-Controller*. Retrieved March 22, 2011, from http://msdn.microsoft.com/en-us/library/ff649643.aspx

Ngo, T. (2011, June). *Standard ECMA-376 Office Open XML File Formats.* Retrieved April 28, 2011, from Ecma International: http://www.ecma-international.org/publications/standards/Ecma-376.htm

O'Reilly & Associates, Inc. (2005). *Chapter 1. The Open Document Format.* Retrieved July 15, 2011, from http://books.evc-cit.info/odbook/ch01.html

OASIS ODF Adoption TC. (2006, December 10). Retrieved July 15, 2011, from OASIS Open: http://www.oasis-open.org/committees/download.php/21450/oasis_odf_advantages_10dec2006.pdf

Pawson, R. (2004). *Naked Objects.* New York: Wiley.

Qiu, X. (2004). *Building Desktop Applications with Web Services in a Message-based MVC Paradigm. Electrical Engineering and Computer Science*, paper 63. http://surface.syr.edu/eecs/63

Sasine, J. M. (1995). Implementing the mdel-view-controller paradigmin ada 95. New York, NY: ACM.

Senior Corps Tech Center. (2008). Retrieved July 2011, from Benefits of Rich Text Format (RTF): http://www.page-house.com/clippings/benefitsOfRTF.html

TechTarget. (2008). *model-view-controller*. Retrieved March 22, 2011, from http://whatis.techtarget.com/definition/0,,sid9_gci214607,00.html

Vugt, W. v. (2007). Open xml the markup explained. In W. v. Vugt, *open x,l the markup explained* (pp. 1–123).

Walther, S. (2008, August 24). *The Evolution of MVC*. Retrieved March 22, 2011, from http://stephenwalther.com/blog/archive/2008/08/24/the-evolution-of-mvc.aspx

Webworld, K. (n.d.). *BAP Objects Design Patterns - Model View Controller MVC I.* Retrieved March 22, 2011, from http://allsapabap.blogspot.com/2009/02/abap-objects-design-patterns-model-view_4191.html

White, E. (2009, Feb 13). Retrieved July 2011, from Seven Key Benefits of Open XML: http://blogs.msdn.com/b/ericwhite/archive/2009/02/13/seven-key-benefits-of-open-xml.aspx

WordIQ. (2010). *Model-view-controller - definition*. Retrieved March 22, 2011, from WordIQ: http://www.wordiq.com/definition/Model-view-controller

Zhengquan, Z., & Chengjun, X. (2010). *MVC model based on high-performance computing operating audit system.* Lanzhou ,Gansu: Second International Conference on MultiMedia and Information Technology.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Doron Drusinsky
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

4. Dr. Bret Michael
   Naval Postgraduate School
   Monterey, California

5. Dr. Man-Tak Shing
   Naval Postgraduate School
   Monterey, California

6. Dr. Thomas Otani
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

7. Mr. John Shea
   Office of the DoD CIO
   Arlington, Virginia